Semaine d'Etude Mathématiques et Entreprises 5 : Reconstruction de couches géologiques à partir de données discrètes

Aurore Back, Mohamed El Bouajaji, Pierre-William Martelli, Brahim Yahiaoui, Jérémy Dalphin, Jonathan Jung, Mario Quillas Saavedra

Semaine d'Etude Maths-Entreprises 5

11-15 Février 2013, Ecole Nationale Supérieure des Mines de Nancy

Reconstruction de couches géologiques à partir de données discrètes

Aurore Back[d] Jérémy Dalphin[a]
Mohamed El Bouajaji[e] Jonathan Jung[b]
Pierre-William Martelli[a] Mario Quillas Saavedra[f]
Brahim Yahiaoui[c]

[a] *Institut Elie Cartan de Lorraine UMR CNRS 7502, Université de Lorraine BP 70239 54506 Vandœuvre-lès-Nancy Cedex, France*
[b] *Institut de Recherche Mathématique Avancée UMR CNRS 7501, 7 rue René Descartes 67084 Strasbourg Cedex, France*
[c] *Institut Français du Pétrole Energies Nouvelles, 1 et 4 avenue de Bois-Préau 92852 Rueil-Malmaison Cedex, France*
[d] *Centre de Physique Théorique UMR CNRS 7332, Campus de Luminy Case 907 13288 Marseille Cedex 9, France*
[e] *Centre de Recherche INRIA Nancy-Grand Est, 615 rue du Jardin Botanique 54600 Villers-lès-Nancy, France*
[f] *Institut de Mathématiques de Jussieu UMR CNRS 7586, 175 rue du Chevaleret 75013 Paris, France*

Sujet proposé par :

Correspondant : Guillaume Caumon (GOCAD consortium)

Résumé

Au cours des Semaines d'Etude Maths-Entreprises, plusieurs jeunes chercheurs en mathématiques se réunissent durant une semaine autour de sujets divers proposés par des entreprises afin d'initier un début de réflexion collective. Dans ce rapport, il s'agit du consortium GOCAD, regroupant de nombreux industriels pétroliers et miniers qui cherchent à reconstituer efficacement le sous-sol terrestre alors qu'ils ne disposent souvent que de données discrètes éparses obtenues lors de forages par exemple.

Un état de l'art est d'abord effectué sur les méthodes récemment développées pour modéliser la géométrie des couches géologiques à partir de points, orientations et lignes. Dans [2], une approche statistique appelée cokrigeage assure une telle reconstruction du sous-sol tandis que dans [3], les données connues sont directement incorporées dans la discrétisation numérique. Finalement, dans [1], un modèle physique est présenté afin d'expliquer les évolutions observées entre deux strates géologiques.

C'est cette dernière modélisation qui aura particulièrement retenu notre attention. La deuxième partie de ce rapport s'efforce donc de l'adapter au problème initial. Chaque couche géologique correspond à un instant t. C'est une surface Γ_t représentée implicitement par les points d'annulation d'une fonction : $\Gamma_t = \{\mathbf{x} \in \mathbb{R}^3,\ \ \Psi(t,\mathbf{x}) = 0\}$. Son évolution au sein du sous-sol est gérée par une équation de type Hamilton-Jacobi : $\frac{\partial \Psi}{\partial t}(t,\mathbf{x}) + H(\mathbf{x}, \vec{\nabla}\Psi(t,\mathbf{x})) = 0$. La loi de propagation H sera de la forme :

$$H\left(\mathbf{x}, \vec{\nabla}\Psi(t,\mathbf{x})\right) = \mu\left(\mathbf{x}\right)\left(\lambda_0(\mathbf{x})\|\vec{\nabla}\Psi(t,\mathbf{x})\| + \sum_{i=1}^{I} \lambda_i(\mathbf{x})\vec{\mathbf{a}}_i.\vec{\nabla}\Psi(t,\mathbf{x})\right)$$

où les λ_i (respectivement μ) paramètreront l'adéquation du modèle par rapport aux orientations connues $\vec{\mathbf{a}}_i$ (respectivement par rapport aux points imposés).

Finalement, on tentera dans un troisième temps de résoudre numériquement les équations de Hamilton-Jacobi associées à cette loi H afin d'obtenir une reconstitution réaliste du sous-sol. Cependant, de nombreux problèmes numériques surviennent lors de la simulation de ce type d'équations. Cela a d'ailleurs donné lieu à une littérature vaste dont un aperçu est donné dans [4]. Par souci de simplicité et surtout par manque de temps, le modèle sera résolu numériquement en 2-D et sans failles.

Mots-clés : couches géologiques, reconstruction, modélisation, Hamilton-Jacobi.

Numéro de publication : SEME005-2013-02-B

Table des matières

Introduction

Récemment mises en place en France, les Semaines d'Etude Mathématiques-Entreprises facilitent et favorisent le rapprochement entre plusieurs jeunes chercheurs, principalement des doctorants en mathématiques, et des entreprises ou compagnies industrielles locales. Ces dernières proposent plusieurs sujets autour desquels les étudiants se réunissent pendant une semaine, l'objectif étant d'entamer un début de réflexion collective dont la pertinence, la qualité et la maturité scientifique, du fait d'années passées en thèse, sont bien souvent supérieures à celles fournies par des ingénieurs tout juste sortis d'Ecole. Le sujet dont il est question dans ce rapport a été initié par le consortium COGAD, regroupant des industriels pétroliers et miniers qui mutualisent leurs coûts d'investissement dans la recherche.

Une connaissance approfondie ainsi qu'une compréhension précise du sous-sol terrestre constituent de véritables enjeux économiques, en particulier au sein de l'industrie pétrolière et minière, parce qu'elles assurent aux entreprises concernées une gestion adéquate des ressources naturelles et des risques associés à leur exploitation. Par exemple, une description fine des couches géologiques permettra à une société productrice de ressources fossiles de savoir où effectuer les coûteux forages servant à l'extraction d'hydrocarbures, à une société de traitement de déchets radioactifs de trouver les zones d'enfouissement les plus sûres, ou encore à des spécialistes de la sécurité minière d'évaluer les risques d'effondrement au sein d'un complexe souterrain de carrières.

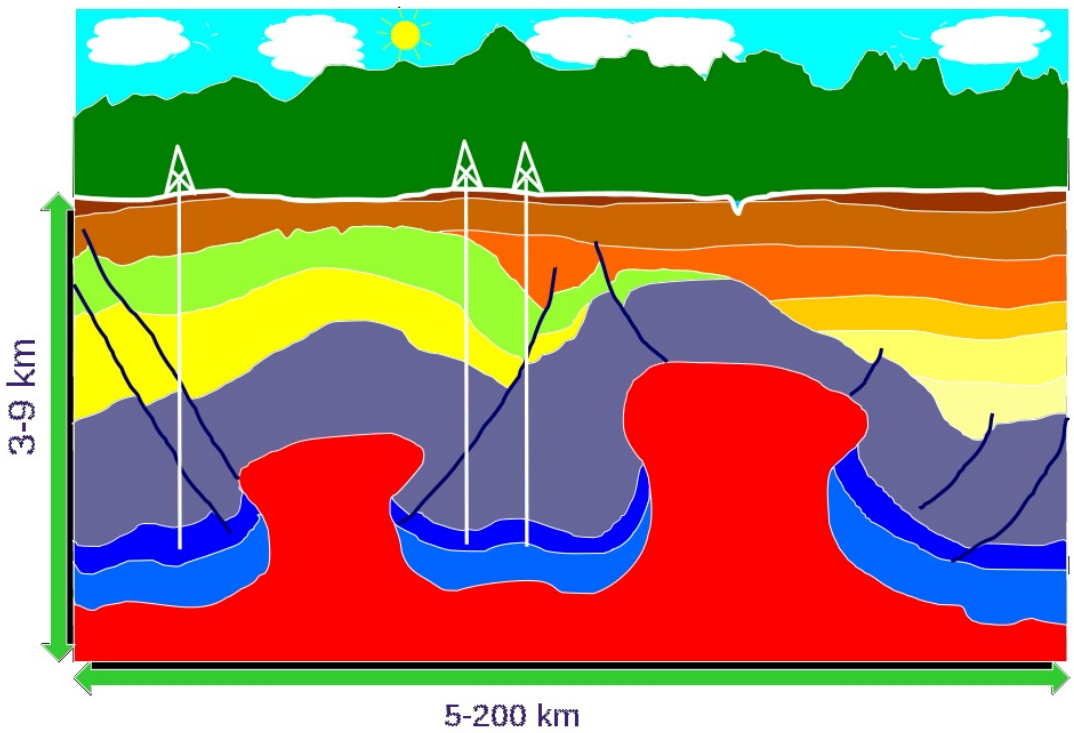

FIGURE 1 – *Exemple de domaine d'étude et des distances associées (source : GOCAD).*

Cependant, notamment en raison du coût élevé des forages, les mesures dont on dispose ne représentent souvent qu'une partie infime et inégalement répartie du sous-sol considéré. De plus, les informations nécessaires pour modéliser la géométrie des couches géologiques sont essentiellement disponibles sous forme discrète : il s'agit de points, d'orientations ou de lignes. Il y a également des failles dont le traitement numérique s'avère assez délicat. On cherche donc ici à représenter le sous-sol en un temps raisonnable à partir d'un nombre important de données éparses. Dans ce rapport, on tentera de préciser les idées qui ont pu émerger au cours de cette semaine de réflexion sur ce sujet vaste et complexe qui passionne ingénieurs et géologues depuis de nombreuses années.

Une modélisation mathématique possible de ce problème consiste à assimiler les strates géologiques aux équipotentielles d'un champ scalaire (cf. Figure 2 pour un tel exemple). Les mesures des forages se traduisent par des contraintes ponctuelles (valeurs imposées) mais aussi vectorielles (directions ou gradients donnés) sur le champ en question. De plus, les informations dont on dispose risquent d'être fortement concentrées sur de petites zones du domaine d'étude, ce qui représente une des difficultés de ce sujet. En effet, il faut être capable de reconstituer le champ scalaire loin des données connues (cf. Figure 1 pour avoir une idée des distances auxquelles on s'intéresse). La seconde complication est liée à la gestion des failles qui induisent une discontinuité des équipotentielles.

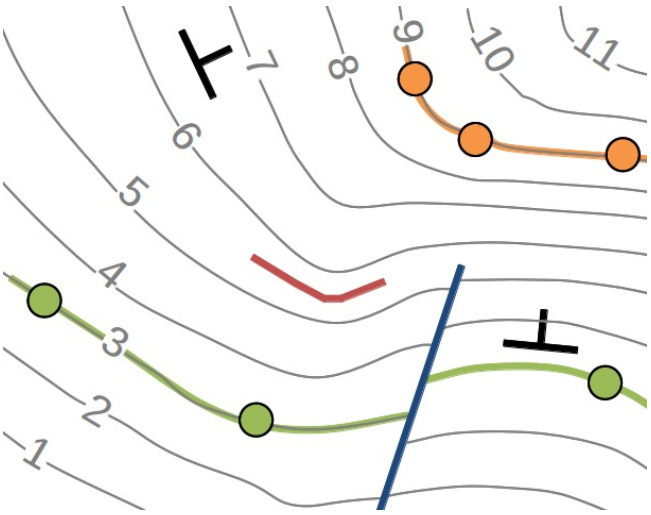

FIGURE 2 – *Exemple de champ scalaire reconstitué (source : Caumon et al. IAMG 2009).*

Une première idée naturelle pour résoudre le problème consiste à interpoler les valeurs discrètes à l'aide d'outils statistiques. Principalement utilisé en géostatistique, le krigeage est une méthode d'estimation linéaire spatiale qui cherche à minimiser la variance entre la moyenne de la fonction aléatoire et celle de son estimateur. Un exemple de ce type de raisonnement est donné dans [2] et semble particulièrement adapté aux données ponctuelles. La technique du cokrigeage y est notamment utilisée afin de pouvoir prendre en compte des gradients et des failles ainsi que la nature géologique du sous-sol. En revanche, le temps de calcul peut être long si le nombre de données est très important et il apparaît parfois des bulles dans la topologie du sous-sol reconstitué, ce qui est généralement impossible.

Une façon de pallier ce problème est donnée dans [3]. L'article propose d'abord de trianguler le domaine spatial dans lequel on travaille, en prenant soin d'adapter le maillage aux failles. Il suffit pour cela de les placer sur le bord de certains tétraèdres. A partir des valeurs et des gradients connus, un système linéaire est ensuite assemblé, en ajoutant aux interfaces des mailles des conditions artificielles de continuité numériquement cohérentes. La dernière étape du raisonnement consiste alors à déterminer une solution de ce système surconditionné par la méthode des moindres carrés. Toutes les données sont ainsi prises en compte et le sous-sol est rapidement reconstitué. Cependant, dans certains cas, la courbure de la surface obtenue n'est pas celle à laquelle s'attendrait un géologue.

Finalement, un point de vue physique peut être adopté en choisissant de modéliser l'accumulation des couches au cours du temps par une équation différentielle évolutive. C'est l'approche effectuée dans [1] : une équation eikonale est utilisée afin de déterminer, pour une couche géologique donnée, la forme de ses voisines proches. Elle nous a semblé particulièrement intéressante et c'est sur celle-ci que nous nous sommes principalement attardés. En effet, elle fournit un modèle théorique simple permettant de retrouver la classification usuelle des divers empilements de strates observés en pratique. Cette méthode a donc l'avantage de pouvoir modéliser localement la géologie des différents types de couches rencontrées dans la nature.

Cependant, la méthode développée dans [1] ne cherche pas a priori à reconstituer le sous-sol dans sa globalité et c'est pourquoi nous nous sommes demandés si nous ne pouvions pas le faire en adaptant la démarche précédente. La seconde partie de ce rapport s'attelle donc à cette tâche. Il s'agit d'incorporer de façon pertinente les données géologiques connues dans la loi d'évolution associée aux équations proposées dans [1]. Le projet esquissé ici présente un avantage : il sépare les compétences et les difficultés du problème, ce qui est très adapté au travail en groupe. L'expertise du géologue sera impliqué dans un choix judicieux de la loi d'évolution tandis que le savoir-faire de l'ingénieur s'appliquera à une résolution numérique efficace des équations de Hamilton-Jacobi obtenues.

La dernière partie de notre travail a donc consisté à traiter numériquement ce nouveau modèle et c'est l'objet de la troisième partie de ce rapport. On y détaille les résultats numériques partiels obtenus à la fin de la semaine ainsi que les nombreuses difficultés qui apparaissent très rapidement malgré le fait que nous nous soyons restreints au cas 2-D et sans failles. L'équation de Hamilton-Jacobi paramétrée par les données est discrétisée à l'aide d'un schéma temporel d'Euler explicite. Le point de vue lagrangien est d'abord naïvement adopté : il s'agit d'effectuer un simple glissement des points de la surface à l'aide des caractéristiques. Puis, une approche eulérienne plus générale est développée comme cela est conseillé dans [4] car elle s'adapte facilement au cas tridimensionnel.

1 Trois méthodes récentes de reconstruction de données

L'objectif est ici de présenter en détail trois méthodes existantes différentes, issues respectivement des récents articles [2], [3], [1], et qui permettent de reconstruire rapidement le sous-sol géologique à partir d'un certain nombre de données discrètes. Dans cette section, on effectue donc un bref état de l'art sur le sujet, en passant successivement en revue une approche statistique, géologique et théorique qui répondent en partie à nos questions.

1.1 Une méthode statistique : le krigeage

Les géologues utilisent souvent les méthodes statistiques pour reconstruire les lignes de niveaux associées à la topographie du sol. La plus classique d'entre elles s'appelle le krigeage. Principalement utilisée en géostatistique, elle permet d'estimer le relief à partir d'un certain nombre de données, en effectuant une interpolation linéaire spatiale sans biais et de variance minimale, pour obtenir un résultat proche de la réalité (cf. Figure 3).

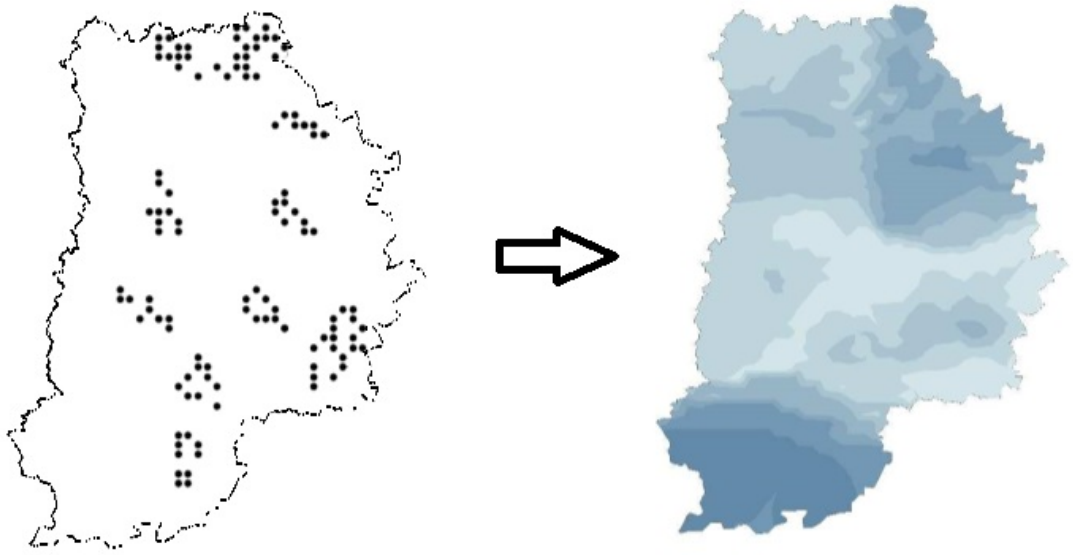

FIGURE 3 – *Simulation de la topographie d'une surface par krigeage (source : Vigie Nature).*

Cependant, en pratique, le calcul effectif par la méthode proprement dite repose sur un certain nombre d'hypothèses qu'il est bon de rappeler et nécessite aussi des informations seulement disponibles au terme d'une étude statistique préalable poussée des données. On commence donc par considérer un échantillon de n données $p_1, \ldots, p_n$ respectivement mesurées aux points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ de l'espace.

1.1.1 Traitement statistique du jeu de données

Tout d'abord, il s'agit d'analyser l'échantillon. Par exemple, l'histogramme de p permet d'évaluer à quel point la distribution s'écarte d'une loi gaussienne. On peut aussi effectuer une étude de la répartition spatiale des valeurs de p afin de détecter la présence de variations systématiques dans les données. Pour cela, il suffit de tracer en chaque point $\mathbf{x}_i$ un cercle dont le rayon correspond à la valeur observée p_i.

Par ailleurs, le tracé d'une nuée variographique permet souvent de rejeter l'une des hypothèses les plus fondamentales du raisonnement qui va suivre, à savoir l'indépendance statistique entre les données. Il est construit à partir d'un nuage de points formé par les couples $(\mathbf{x}_i - \mathbf{x}_j, (p_i - p_j)^2)$ pour tout $(i,j) \in \{1,\dots,n\}^2$. Ce phénomène de non-indépendance est un principe aussi appelé auto-corrélation spatiale en géographie.

Ensuite, on cherche à modéliser la distribution statistique du jeu de données $p_1,\dots,p_n$. Dans le cas où les $p_1,\dots,p_n$ sont supposés être les réalisations respectives de variables aléatoires $T_1,\dots,T_n$ indépendantes et identiquement distribuées, il suffit de spécifier la fonction de répartition ou la loi qu'elles partagent en commun, et sinon, il faut préciser celle de chaque T_i, de chaque couple (T_i, T_j), de chaque triplet, etc.

Plus généralement, on a noté p la valeur de la variable statistique au point $\mathbf{x}$ et on suppose qu'elle est la réalisation d'une variable aléatoire $T_{\mathbf{x}}$. L'application $T : \mathbf{x} \mapsto T_{\mathbf{x}}$ est alors appelée fonction aléatoire. Sa distribution statistique, dont l'estimation reste à faire, est souvent déterminée à l'aide de ses moments d'ordre un et deux, qui se définissent en tout point $\mathbf{x}$, $\tilde{\mathbf{x}}$ de l'espace par les formules suivantes :

$$\begin{cases} \mu(\mathbf{x}) = \mathbb{E}\left(T_{\mathbf{x}}\right) \\ \\ C(\mathbf{x},\tilde{\mathbf{x}}) = \mathrm{Cov}\left(T_{\mathbf{x}}, T_{\tilde{\mathbf{x}}}\right) = \mathbb{E}\left[\left(T_{\mathbf{x}} - \mu(\mathbf{x})\right)\left(T_{\tilde{\mathbf{x}}} - \mu(\tilde{\mathbf{x}})\right)\right] = \mathbb{E}\left(T_{\mathbf{x}} T_{\tilde{\mathbf{x}}}\right) - \mu(\mathbf{x})\mu(\tilde{\mathbf{x}}). \end{cases}$$

On dit que la fonction aléatoire T est stationnaire lorsque l'application $\mu : \mathbf{x} \mapsto \mu(\mathbf{x})$ est constante et si $C : (\mathbf{x},\tilde{\mathbf{x}}) \mapsto C(\mathbf{x},\tilde{\mathbf{x}})$ peut être mis sous la forme $C(\mathbf{x},\tilde{\mathbf{x}}) = C(\|\mathbf{x} - \tilde{\mathbf{x}}\|)$. Il se trouve que l'hypothèse de stationnarité constitue en première approximation un bon compromis entre exactitude des données et simplicité mathématique. Mais bien souvent, elle est faite d'emblée car il est difficile de la tester efficacement ou de faire autrement.

Finalement, sous cette hypothèse de stationnarité des variables aléatoires $T_{\mathbf{x}}$, il ne reste donc plus qu'à estimer la fonction $C : h \mapsto C(h)$. Celle-ci peut notamment être inférée à partir du variogramme ou simplement grâce au tracé de la covariance empirique $C^*(h) = \frac{1}{n_h}\sum_{(i,j)\in\mathcal{S}_h} p_i p_j - \bar{p}$, où $\bar{p} = \sum_{i=1}^n p_i$ et n_h désigne le cardinal de l'ensemble $\mathcal{S}_h = \{(i,j) \in \{1,\dots,n\}^2, \quad |p_i - p_j| \approx h\}$.

L'ajustement de C peut être fait à la main, technique la plus simple à mettre en œuvre. On utilise aussi très souvent un modèle paramétrique (pépitique, exponentiel ou gaussien) en sélectionnant la valeur du paramètre qui minimise la somme des erreurs quadratiques. Il est également possible de maximiser la vraisemblance pour pouvoir estimer la valeur de ce paramètre.

1.1.2 Le krigeage proprement dit

Une fois la fonction $C : h \mapsto C(h)$ correctement estimée, la méthode du krigeage peut alors être mise en place pour interpoler les valeurs de p à partir des données $p_1, \ldots, p_n$. On considère donc un certain point $\mathbf{x}$ pour lequel on souhaite estimer $T_\mathbf{x}$ à l'aide d'un estimateur statistique noté $T^*_\mathbf{x}$. Il est supposé être linéaire et sans biais :

$$\begin{cases} \forall i \in \{1, \ldots, n\}, \quad T^*_{\mathbf{x}_i} = T_{\mathbf{x}_i} = T_i \\ \\ \mathbb{E}\left(T^*_\mathbf{x}\right) = \mathbb{E}\left(T_\mathbf{x}\right) = \mu(\mathbf{x}) = m \\ \\ \exists (\lambda^1_\mathbf{x}, \ldots, \lambda^n_\mathbf{x}) \in \mathbb{R}^n, \quad T^*_\mathbf{x} = \displaystyle\sum_{i=1}^n \lambda^i_\mathbf{x} T_i. \end{cases}$$

Le krigeage consiste alors à déterminer les coefficients $\lambda^1_\mathbf{x}, \ldots, \lambda^n_\mathbf{x}$ qui minimisent l'erreur quadratique commise entre $T^*_\mathbf{x}$ et $T_\mathbf{x}$, c'est-à-dire ceux qui rendent minimale la variance :

$$\begin{aligned} \mathrm{Var}\left(T^*_\mathbf{x} - T_\mathbf{x}\right) &= \mathrm{Var}\left(T_\mathbf{x}\right) + \sum_{i=1}^n \sum_{j=1}^n \lambda^i_\mathbf{x} \lambda^j_\mathbf{x} \mathrm{Cov}\left(T_i, T_j\right) - 2 \sum_{i=1}^n \lambda^i_\mathbf{x} \mathrm{Cov}\left(T_\mathbf{x}, T_i\right) \\ &= C(0) + \sum_{i=1}^n \sum_{j=1}^n \lambda^i_\mathbf{x} \lambda^j_\mathbf{x} \underbrace{C\left(\|\mathbf{x}_i - \mathbf{x}_j\|\right)}_{:= C_{i,j}} - 2 \sum_{i=1}^n \lambda^i_\mathbf{x} \underbrace{C\left(\|\mathbf{x} - \mathbf{x}_i\|\right)}_{:= C^i_\mathbf{x}}. \end{aligned}$$

Il s'agit d'un problème classique d'optimisation linéaire qui se résout en étudiant les points critiques du lagrangien associé, en prenant soin de distinguer le cas du krigeage simple où la moyenne $\mu = m$ est connue de celui où elle ne l'est pas, appelé krigeage ordinaire.

Dans la première situation, quitte à remplacer la fonction aléatoire T par $T - m$, on peut supposer que $m = 0$. Il en résulte que l'estimateur T^* est sans biais puisqu'on a :

$$\mathbb{E}\left(T^*_\mathbf{x}\right) = \mathbb{E}\left(T_\mathbf{x}\right) \quad \Longleftrightarrow \quad m\left(1 - \sum_{i=1}^n \lambda^i_\mathbf{x}\right) = 0.$$

Les conditions d'optimalité du problème $\min_{(\lambda^1_\mathbf{x}, \ldots, \lambda^n_\mathbf{x}) \in \mathbb{R}^n} \mathrm{Var}(T^*_\mathbf{x} - T_\mathbf{x})$ s'écrivent alors tout simplement sous la forme matricielle suivante :

$$\begin{pmatrix} C_{1,1} & \cdots & C_{1,n} \\ \vdots & \ddots & \vdots \\ C_{n,1} & \cdots & C_{n,n} \end{pmatrix} \cdot \begin{pmatrix} \lambda^1_\mathbf{x} \\ \vdots \\ \lambda^n_\mathbf{x} \end{pmatrix} = \begin{pmatrix} C^1_\mathbf{x} \\ \vdots \\ C^n_\mathbf{x} \end{pmatrix}$$

et c'est la résolution de ce système linéaire qui permet d'obtenir les coefficients $\lambda^1_\mathbf{x}, \ldots, \lambda^n_\mathbf{x}$ assurant à la variance d'être minimale. Cependant, comme la matrice $(C_{i,j})_{1 \leqslant i,j \leqslant n}$ est pleine en général, on résout le système sans calculer l'inverse, ce qui prendrait trop de temps. En pratique, le krigeage ne prend en compte que les plus proches voisins, les voisins distants ayant peu d'influence sur le résultat. C'est dans le cas du krigeage dual (voisinage global) que le problème se pose car il faut alors explicitement inverser la matrice.

Par ailleurs, dans la seconde situation où m n'est pas connu, la contrainte $\sum_{i=1}^{n} \lambda_{\mathbf{x}}^{i} = 1$ doit être imposée afin que l'estimateur $T_{\mathbf{x}}^{*}$ soit bien sans biais. L'écriture des conditions d'optimalité fait donc apparaître cette fois un multiplicateur de Lagrange ξ et le système linéaire à résoudre devient :

$$\begin{pmatrix} C_{1,1} & \cdots & C_{1,n} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ C_{n,1} & \cdots & C_{n,n} & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda_{\mathbf{x}}^{1} \\ \vdots \\ \lambda_{\mathbf{x}}^{n} \\ \xi \end{pmatrix} = \begin{pmatrix} C_{\mathbf{x}}^{1} \\ \vdots \\ C_{\mathbf{x}}^{n} \\ 1 \end{pmatrix}.$$

1.1.3 Le cokrigeage

De manière plus générale, il est possible de rajouter d'autres types d'informations. Dans [2], l'approche du cokrigeage est développée afin de pouvoir prendre en compte des directions d'orientation des couches et ainsi mieux contrôler l'interpolation finale obtenue. La méthode du krigeage est ainsi généralisée à deux fonctions aléatoires (cf. Figure 4).

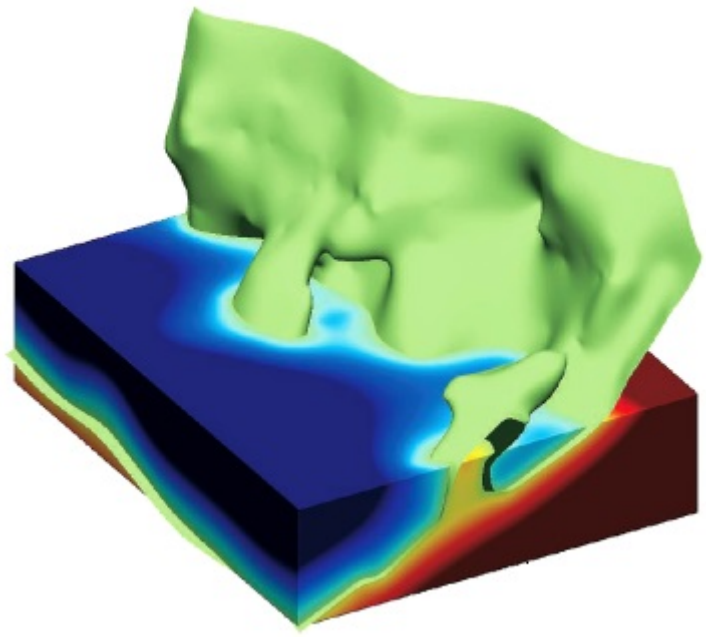

FIGURE 4 – *Exemple de reconstruction 3-D par cokrigeage (source : GOCAD).*

On suppose donc disposer d'un échantillon supplémentaire de l vecteurs notés $\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_l$ respectivement situés aux points $\mathbf{y}_1, \ldots, \mathbf{y}_l$. On définit alors la dérivée directionnelle de la fonction aléatoire T dans la direction $\vec{\mathbf{a}}_j$ au point $\mathbf{x}$ par la formule $\partial_{\vec{\mathbf{a}}_j} T_{\mathbf{x}} = \vec{\nabla} T(\mathbf{x}).\vec{\mathbf{a}}_j$. Puis, on choisit un estimateur de $T_{\mathbf{x}}$ linéaire et sans biais de la forme :

$$T_{\mathbf{x}}^{*} = \sum_{i=1}^{n} \lambda_{\mathbf{x}}^{i} T_{\mathbf{x}_i} + \sum_{j=1}^{l} \nu_{\mathbf{x}}^{j} \partial_{\vec{\mathbf{a}}_j} T_{\mathbf{y}_j}.$$

Les coefficients $\lambda_{\mathbf{x}}^1, \ldots, \lambda_{\mathbf{x}}^n$ et $\nu_{\mathbf{x}}^1, \ldots, \nu_{\mathbf{x}}^l$ sont ceux qui minimisent la variance de $T_{\mathbf{x}}^* - T_{\mathbf{x}}$, l'écriture des conditions d'optimalité amenant à la résolution d'un gros système linéaire. Comme précédemment, l'inversion d'une matrice pleine est très coûteuse en temps ce qui limite l'utilisation du cokrigeage à un nombre restreint de données.

Finalement, dans [2], la fonction de répartition associée à T vaut $f(\mathbf{x}) = \sum_{k=1}^{K} b_k g^k(\mathbf{x})$, les b_k et g^k étant choisis de façon à contrôler au mieux la propagation de fonctions radiales. Un polynôme est généralement la forme retenue pour f et dans [2], il est démontré qu'alors, la fonction de répartition associée à $\partial_{\vec{\mathbf{a}}_j} T_{\mathbf{y}_j}$ est bien $\partial_{\vec{\mathbf{a}}_j} f(\mathbf{y}_j) = \vec{\nabla} f(\mathbf{y}_j).\vec{\mathbf{a}}_j$.

En pratique, cette méthode est acceptable pour un échantillon raisonnable de données. La représentation des failles reste quant à elle limitée, même s'il est possible de choisir des fonctions de répartition discontinues afin de mieux les modéliser. Il est aussi possible de représenter les érosions partielles des couches sédimentaires par minimisation discrète.

1.2 Une autre approche géologique et numérique

Tout d'abord, on rappelle le problème dont il est ici question : il s'agit de rapidement reconstruire une partie $\Omega \subseteq \mathbb{R}^3$ du sous-sol terrestre à partir d'un certain nombre de données. Le domaine Ω considéré est formé de diverses couches géologiques associées à différents instants t. Plus précisément, ce sont des surfaces Γ_t qu'on décide de représenter comme les équipotentielles d'un champ scalaire $T : \Omega \to \mathbb{R}$. On a donc :

$$\Gamma_t = \{\mathbf{x} \in \Omega, \quad T(\mathbf{x}) = t\}.$$

Les mesures dont on dispose pour reconstruire la fonction T se traduisent par des contraintes ponctuelles, c'est-à-dire la donnée de I valeurs de T imposées, notées $(T_i)_{1 \leqslant i \leqslant I}$ et respectivement mesurées aux points $(\mathbf{x}_i)_{1 \leqslant i \leqslant I} = (x_i, y_i, z_i)_{1 \leqslant i \leqslant I}$, mais aussi par des contraintes directionnelles, i.e. les J vecteurs $(\vec{\mathbf{a}}_j)_{1 \leqslant j \leqslant J}$ correspondant à la valeur de $\vec{\nabla} T$ aux points $(\mathbf{y}_j)_{1 \leqslant j \leqslant J} = (x_j, y_j, z_j)_{1 \leqslant j \leqslant J}$.

Ensuite, la démarche suivie dans [3] consiste à discrétiser Ω de manière à bien prendre en compte les failles, puis d'assembler un système linéaire à partir des $I + J$ contraintes, de telle sorte que la matrice soit creuse, réduisant ainsi considérablement le temps de calcul nécessaire à la reconstruction de Ω. On détaille maintenant comment discrétiser convenablement le domaine.

Le domaine Ω est découpé en plusieurs tétraèdres $(\mathcal{T}_l)_{1 \leqslant l \leqslant L}$ de façon à ce qu'ils forment un maillage conforme de Ω. Désormais, les inconnues du système discrétisé sont donc les valeurs de T aux quatre sommets de chaque tétraèdre de notre maillage. On les note T_l^k pour tout entier $l \in \{1, \ldots, L\}$ et tout $k \in \{1, 2, 3, 4\}$. On fait alors l'hypothèse que T est une fonction linéaire sur chaque tétraèdre $\mathcal{T}_l$.

De plus, on suppose que le maillage a été réalisé de manière à ce que les failles se retrouvent forcément sur le bord de certains tétraèdres. Nous allons maintenant expliquer comment prendre en compte les valeurs connues de la fonction T et de son gradient $\vec{\nabla}T$ dans la discrétisation du domaine Ω qu'on vient d'effectuer. Plusieurs situations peuvent alors se présenter.

1.2.1 Utilisation des points $\mathbf{x}_i$ où la valeur de T est connue et vaut T_i

Soit un entier $i \in \{1, \ldots, I\}$ fixé. On commence par localiser parmi les L tétraèdres, celui pour lequel on a :

$$\mathbf{x}_i \in \mathcal{T}_l.$$

Comme la fonction T a été supposée linéaire sur chaque tétraèdre, on peut exprimer la valeur T_i de T au point $\mathbf{x}_i$ à partir des valeurs $(T_l^k)_{1 \leqslant k \leqslant 4}$ de T aux quatre sommets de $\mathcal{T}_l$. Il suffit pour cela de calculer les coordonnées barycentriques $(u_l^k)_{1 \leqslant k \leqslant 4}$ du point $\mathbf{x}_i$ dans le tétraèdre $\mathcal{T}_l$. Elles sont obtenues en résolvant le système linéaire suivant :

$$\begin{pmatrix} x_l^1 & x_l^2 & x_l^3 & x_l^4 \\ y_l^1 & y_l^2 & y_l^3 & y_l^4 \\ z_l^1 & z_l^2 & z_l^3 & z_l^4 \\ 1 & 1 & 1 & 1 \end{pmatrix} . \begin{pmatrix} u_l^1 \\ u_l^2 \\ u_l^3 \\ u_l^4 \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

où $(\mathbf{x}_l^k)_{1 \leqslant k \leqslant 4} = (x_l^k, y_l^k, z_l^k)_{1 \leqslant k \leqslant 4}$ désignent les points associés aux quatre sommets de $\mathcal{T}_l$. On écrit alors :

$$\sum_{k=1}^{4} u_l^k T_l^k = T_i$$

ce qui constitue bien une équation linéaire faisant intervenir les inconnues T_l^k.

1.2.2 Utilisation des points $\mathbf{y}_j$ où la valeur de $\vec{\nabla}T$ est connu et vaut $\vec{\mathbf{a}}_j$

Comme précédemment, on localise le tétraèdre $\mathcal{T}_l$ pour lequel on a $\mathbf{y}_j \in \mathcal{T}_l$. Par linéarité de la fonction T sur $\mathcal{T}_l$, son gradient $\vec{\nabla}T$ y est constant. On peut alors exprimer la contrainte vectorielle sous la forme matricielle suivante :

$$\begin{pmatrix} x_l^2 - x_l^1 & y_l^2 - y_l^1 & z_l^2 - z_l^1 \\ x_l^3 - x_l^1 & y_l^3 - y_l^1 & z_l^3 - z_l^1 \\ x_l^4 - x_1^1 & y_l^4 - y_l^1 & z_l^4 - z_l^1 \end{pmatrix}^{-1} . \begin{pmatrix} T_l^2 - T_l^1 \\ T_l^3 - T_l^1 \\ T_l^4 - T_l^1 \end{pmatrix} = \vec{\mathbf{a}}_j$$

où les $(\mathbf{x}_l^k)_{1 \leqslant k \leqslant 4} = (x_l^k, y_l^k, z_l^k)_{1 \leqslant k \leqslant 4}$ désignent les coordonnées des quatre sommets de $\mathcal{T}_l$. Par conséquent, cela fournit trois équations linéaires pour chaque entier $j \in \{1, \ldots, J\}$ faisant intervenir les inconnues T_l^k. Il convient maintenant de rajouter un certain nombre de conditions supplémentaires afin d'obtenir suffisamment d'équations dans le système. Elles traduiront la continuité de T aux sommets ainsi que celle de la composante normale de $\vec{\nabla}T$ aux interfaces de chaque tétraèdre.

1.2.3 Conditions imposées à T sur les sommets communs à deux tétraèdres

Lorsque deux tétraèdres $\mathcal{T}_{l_1}$ et $\mathcal{T}_{l_2}$ ont un sommet $\mathbf{x}$ en commun qui ne se situe pas sur une faille, on suppose que les valeurs de T sont identiques en ce point $\mathbf{x}$. Autrement dit, s'il existe $(k_1, k_2) \in \{1,2,3,4\}^2$ tel que $\mathbf{x}_{l_1}^{k_1} = \mathbf{x}_{l_2}^{k_2}$, alors on impose $T_{l_1}^{k_1} = T_{l_2}^{k_2}$ où on rappelle que $(\mathbf{x}_l^k)_{1 \leqslant k \leqslant 4}$ désigne les quatre sommets de $\mathcal{T}_l$. De plus, cette approche nous permet de traiter les failles en dédoublant simplement les valeurs de T de part et d'autre de celles-ci.

1.2.4 Conditions imposées à $\vec{\nabla} T$ sur l'interface commune à deux tétraèdres

Quand deux tétraèdres $\mathcal{T}_{l_1}$ et $\mathcal{T}_{l_2}$ ont en commun une surface $\mathcal{S}$ ne se situant pas sur une faille, on fait l'hypothèse que la dérivée normale de T est la même i.e. continue sur $\mathcal{S}$. En notant $\vec{\mathbf{n}}_{\mathcal{S}}$ un vecteur normal unitaire à la surface $\mathcal{S}$, cela signifie que :

$$\begin{pmatrix} x_{l_1}^2 - x_{l_1}^1 & y_{l_1}^2 - y_{l_1}^1 & z_{l_1}^2 - z_{l_1}^1 \\ x_{l_1}^3 - x_{l_1}^1 & y_{l_1}^3 - y_{l_1}^1 & z_{l_1}^3 - z_{l_1}^1 \\ x_{l_1}^4 - x_{l_1}^1 & y_{l_1}^4 - y_{l_1}^1 & z_{l_1}^4 - z_{l_1}^1 \end{pmatrix}^{-1} \begin{pmatrix} T_{l_1}^2 - T_{l_1}^1 \\ T_{l_1}^3 - T_{l_1}^1 \\ T_{l_1}^4 - T_{l_1}^1 \end{pmatrix} . \vec{\mathbf{n}}_{\mathcal{S}}$$

$$= \begin{pmatrix} x_{l_2}^2 - x_{l_2}^1 & y_{l_2}^2 - y_{l_2}^1 & z_{l_2}^2 - z_{l_2}^1 \\ x_{l_2}^3 - x_{l_2}^1 & y_{l_2}^3 - y_{l_2}^1 & z_{l_2}^3 - z_{l_2}^1 \\ x_{l_2}^4 - x_{l_2}^1 & y_{l_2}^4 - y_{l_2}^1 & z_{l_2}^4 - z_{l_2}^1 \end{pmatrix}^{-1} \begin{pmatrix} T_{l_2}^2 - T_{l_2}^1 \\ T_{l_2}^3 - T_{l_2}^1 \\ T_{l_2}^4 - T_{l_2}^1 \end{pmatrix} . \vec{\mathbf{n}}_{\mathcal{S}}.$$

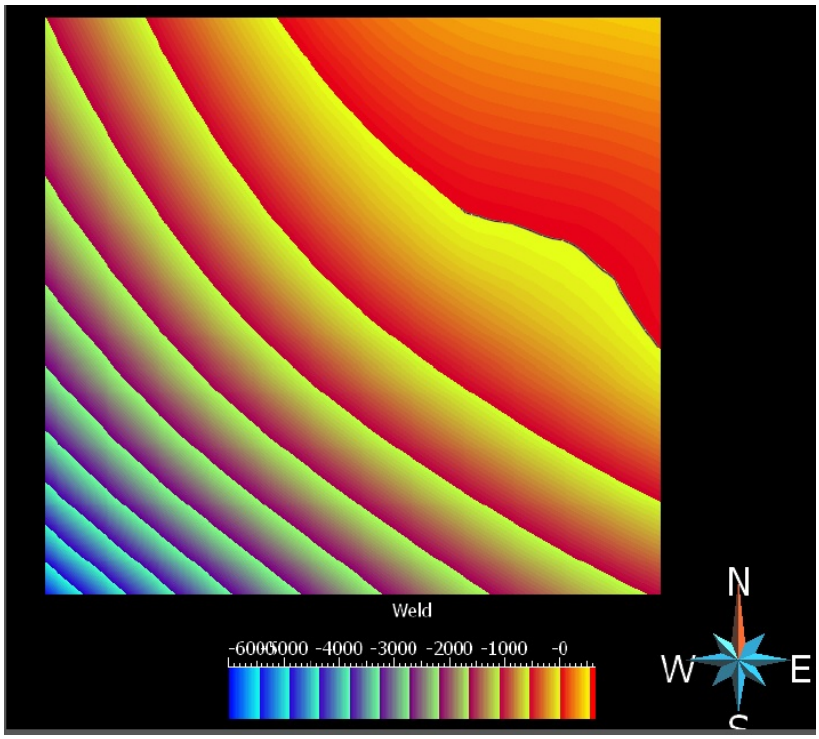

FIGURE 5 – *Exemple de champ scalaire correspondant à une faille (source : GOCAD).*

1.2.5 Assemblage et résolution du système linéaire obtenu

Les équations des quatre derniers paragraphes constituent bien un système linéaire à résoudre ayant pour inconnues les T_l^k. En pratique, il est sur-contraint et on décide de choisir la meilleure solution au sens des moindres carrés, c'est-à-dire celle qui minimise l'erreur quadratique entre les valeurs données et la solution obtenue. On obtient ainsi une reconstitution rapide et satisfaisante du sous-sol car la matrice du système considéré est creuse.

Toutefois, cette méthode a ses limites car elle prend seulement en compte des contraintes locales d'ordre un. En effet, comme on peut l'observer assez clairement sur la Figure 5, on obtient des lignes de niveaux qui n'ont pas la courbure à laquelle s'attendrait un géologue. Elles devraient être concaves plutôt que convexes. Ceci vient de la contrainte de continuité du gradient (cf. Section 1.2.4) qui empêche de pouvoir imposer une condition de continuité sur la courbure par exemple.

1.3 Une démarche plus théorique

On étudie maintenant l'approche développée dans [1]. L'objectif de cet article est de modéliser par une loi physique l'ensemble des géométries de couches plissées qui ont été classifiées par Ramsay en 1967. Ce dernier distingue cinq types de géométrie locale de plis (cf. Figure 6) : il s'agit des classes 1A, 1B, 1C, 2 et 3. Elles sont essentiellement obtenues en mesurant la variation d'épaisseur le long de la direction des isogones (lignes reliant les points de même inclinaison).

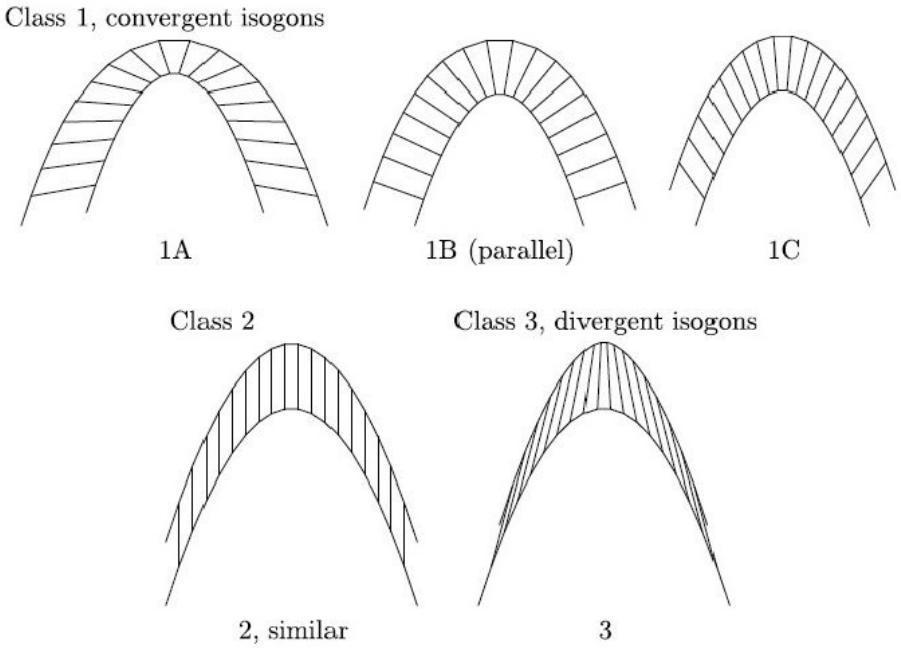

FIGURE 6 – *Les 5 types de plis définis par Ramsay et leurs isogones (source : issu de [1]).*

Plus précisément, étant donné un certain angle d'inclinaison noté α, on rappelle qu'une isogone est le segment qui connecte les deux points de deux couches géologiques distinctes par lesquels la tangente, également appelée isoligne, forme un angle α avec l'horizontale (cf. Figure 7). On obtient alors la classification de Ramsay en comparant les différentes distances t_α formées par les isolignes. La Figure 8 représente l'épaisseur adimensionnée $t'_\alpha = \frac{t_\alpha}{t_0}$ en fonction de α, ce qui permet d'apprécier les différents types de classes possibles.

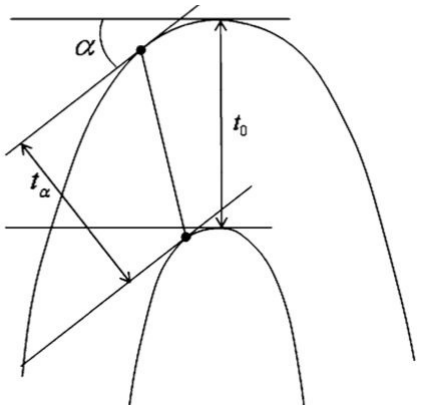

FIGURE 7 – *Schéma d'une isogone (source : issu de [1]).*

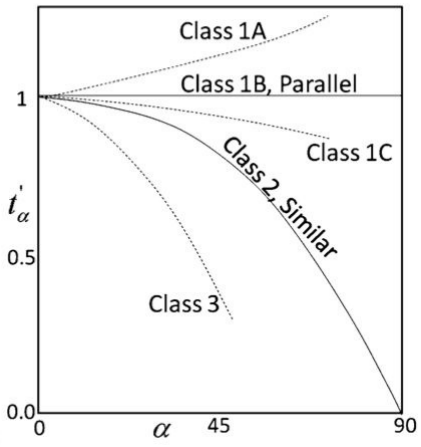

FIGURE 8 – *Variation adimensionnelle de la longueur des isogones en fonction de l'angle d'inclinaison (source : issu de [1]).*

1.3.1 La modélisation mathématique

On commence par se donner une surface Γ évoluant dans l'espace tridimensionnel $\mathbb{R}^3$. On suppose que chaque point $\mathbf{x}$ appartenant à Γ se déplace dans la direction normale, modélisée par le vecteur $\vec{\mathbf{n}}(\mathbf{x})$, à une certaine vitesse notée $F(\mathbf{x}, \vec{\mathbf{n}})$. Afin que les couches ne puissent se croiser, on impose à F d'être toujours positive. Puis, on considère le temps d'arrivée $T(\mathbf{x})$ de la surface Γ au point $\mathbf{x}$. Les couches Γ_t correspondant à l'instant t sont alors les lignes de niveaux de T, c'est-à-dire que $\Gamma_t = \{\mathbf{x} \in \mathbb{R}^3, \ \ T(\mathbf{x}) = t\}$.

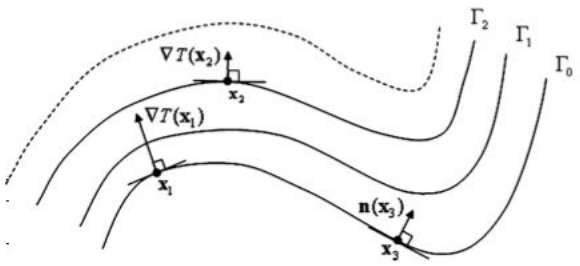

FIGURE 9 – *Déplacement des plis le long d'un front de propagation (source : issu de [1]).*

Comme le vecteur normal $\vec{\mathbf{n}}(\mathbf{x})$ est unitaire et qu'il a la même direction, le même sens que le gradient $\vec{\nabla}T(\mathbf{x})$ de T au point $\mathbf{x}$, il s'exprime donc sous la forme (cf. Figure 9) :

$$\vec{\mathbf{n}}(\mathbf{x}) = \frac{\vec{\nabla}T(\mathbf{x})}{\|\vec{\nabla}T(\mathbf{x})\|}.$$

Si on regarde localement comment évoluent les surfaces, on remarque que $\mathbf{x}$ se déplace vers le point $\mathbf{x} + \Delta t\ F(\mathbf{x}, \vec{\mathbf{n}})\vec{\mathbf{n}}$ entre les instants t et $t + \Delta t$. Il en résulte que :

$$T\left(\mathbf{x} + \Delta t\ F(\mathbf{x}, \vec{\mathbf{n}})\vec{\mathbf{n}}\right) = t + \Delta t.$$

On effectue alors un développement limité au premier ordre de la fonction T autour de $\mathbf{x}$ et il vient la relation suivante :

$$T(\mathbf{x}) + \Delta t\ F(\mathbf{x}, \vec{\mathbf{n}}) \left(\vec{\nabla}T(\mathbf{x}).\vec{\mathbf{n}}\right) \approx t + \Delta t.$$

On rappelle que $T(\mathbf{x}) = t$ et en utilisant l'expression de $\vec{\mathbf{n}}(\mathbf{x})$ précédente, on obtient ainsi l'équation d'évolution recherchée :

$$\|\vec{\nabla}T(\mathbf{x})\| F\left(\mathbf{x}, \frac{\vec{\nabla}T(\mathbf{x})}{\|\vec{\nabla}T(\mathbf{x})\|}\right) = 1.$$

Il s'agit d'une équation eikonale non-linéaire appartenant à la classe des équations de Hamilton-Jacobi. En introduisant le Hamiltonien $H(\mathbf{x}, \vec{\nabla}T(\mathbf{x})) = \|\vec{\nabla}T(\mathbf{x})\| F(\mathbf{x}, \vec{\mathbf{n}}(\mathbf{x}))$, l'équation s'écrit alors simplement $H(\mathbf{x}, \vec{\nabla}T(\mathbf{x})) = 1$. Elle modélise un phénomène de propagation : elle est donc de type hyperbolique, comme l'équation des ondes. Afin d'alléger les notations, on note dorénavant $\vec{\mathbf{p}} = \vec{\nabla}T(\mathbf{x})$. Ce sera la seconde variable de H et elle sera supposée indépendante de la première, à savoir $\mathbf{x}$.

Une façon classique d'affronter ce type d'équations s'effectue par la méthode dite des caractéristiques. Cela consiste à étudier les courbes paramétrées $t \mapsto \mathbf{x}(t)$ et $t \mapsto \vec{\mathbf{p}}(t)$ le long desquelles H reste constant. On a donc :

$$\begin{cases} H\left(\mathbf{x}(t), \vec{\mathbf{p}}(t)\right) = 1 \\ \\ T\left(\mathbf{x}(t)\right) = t \end{cases}$$

et en dérivant par rapport au paramètre t, on obtient :

$$\begin{cases} \dot{\mathbf{x}}(t).\vec{\nabla}_{\mathbf{x}} H\left(\mathbf{x}(t), \vec{\mathbf{p}}(t)\right) + \dot{\vec{\mathbf{p}}}(t).\vec{\nabla}_{\vec{\mathbf{p}}} H\left(\mathbf{x}(t), \vec{\mathbf{p}}(t)\right) = 0 \\ \\ \dot{\mathbf{x}}(t).\vec{\nabla}T(\mathbf{x}) = 1. \end{cases}$$

Finalement, la première équation du système précédent est vérifiée quand on identifie $\dot{\mathbf{x}}$ avec $\vec{\nabla}_{\vec{\mathbf{p}}} H$ et $\dot{\vec{\mathbf{p}}}$ avec $-\vec{\nabla}_{\mathbf{x}} H$. On a ainsi transformé une équation aux dérivées partielles en un système d'équations différentielles ordinaires vérifiées sur les caractéristiques :

$$\begin{cases} \dot{\mathbf{x}} = \vec{\nabla}_{\vec{\mathbf{p}}} H \\ \\ \dot{\vec{\mathbf{p}}} = -\vec{\nabla}_{\mathbf{x}} H \\ \\ \dot{\mathbf{x}}.\vec{\mathbf{p}} = 1. \end{cases}$$

1.3.2 Retrouver la classification de Ramsay à partir du modèle

On rappelle que l'objectif est ici d'obtenir un modèle mathématique qui nous permette de retrouver les cinq types de plissement donnés par Ramsay en 1967. Grâce à l'approche précédente, il nous suffit donc de seulement connaître le champ de vitesse $F(\mathbf{x}, \vec{\mathbf{n}})$ pour résoudre le système précédent et ainsi obtenir la fonction T qui contient, grâce à ses lignes de niveaux, les informations nécessaires à la reconstruction du sous-sol. On verra plus tard que numériquement, ceci n'est pas du tout une tâche facile.

A partir du modèle, il est montré dans [1] que les isogones sont exactement les courbes caractéristiques de l'équation. Ainsi, quand on examine la classe 1B, on s'aperçoit que le front de propagation se situe seulement le long de la direction normale. Dans ce cas, on se donne donc un champ de vitesse sous la forme $F(\mathbf{x}, \vec{\mathbf{n}}) = F_{\text{prop}}(\mathbf{x})$, i.e. qui dépend uniquement de la position $\mathbf{x}$ et non de la normale $\vec{\mathbf{n}}$. En effet, dans le plissement 1B, il n'y a aucune variation d'épaisseur entre deux couches lors du déplacement selon la normale.

Si on analyse maintenant le plissement de type 2, on remarque qu'il n'y a pas non plus de variation d'épaisseur entre deux couches, mais maintenant, celle-ci est mesurée dans une direction privilégiée, qu'on va représenter par un vecteur unitaire noté $\vec{\mathbf{a}}$. Autrement dit, la forme du champ de vitesse ne va pas être exprimée par un terme de propagation comme pour le type 1B, mais plutôt par un terme d'advection pur dans la direction $\vec{\mathbf{a}}$, ce qui peut se traduire par le fait que $F(\mathbf{x}, \vec{\mathbf{n}}) = F_{\text{adv}}(\mathbf{x}, \vec{\mathbf{n}}) = \psi_{\text{adv}}(\mathbf{x})(\vec{\mathbf{a}}.\vec{\mathbf{n}})$.

Comme la vitesse dépend ici de la direction de propagation, c'est-à-dire de $\vec{\mathbf{a}}$, on parle d'anisotropie, contrairement au plissement de type 1B pour lequel le front de propagation est qualifié d'isotrope. Concernant les autres types de plissement, ils se modélisent en mélangeant ces deux situations, c'est-à-dire que le champ de vitesse sera construit avec des contributions liées à une propagation normale et à un terme d'advection :

$$F(\mathbf{x}, \vec{\mathbf{n}}) = F_{\text{prop}}(\mathbf{x}) + F_{\text{adv}}(\mathbf{x}, \vec{\mathbf{n}}) = F_{\text{prop}}(\mathbf{x}) + \psi_{\text{adv}}(\mathbf{x})\,(\vec{\mathbf{a}}.\vec{\mathbf{n}})\,.$$

Pour conclure, le tableau qui suit donne la classification de Ramsay en fonction du signe des fonctions F_{prop} et ψ_{adv} lorsqu'elles sont supposées constantes.

Type de Plissement	F_{prop}	ψ_{adv}
Classe 1A	> 0	< 0
Classe 1B	> 0	$= 0$
Classe 1C	> 0	> 0
Classe 2	$= 0$	> 0
Classe 3	< 0	> 0

2 Comment reconstituer le sous-sol à partir d'une équation de Hamilton-Jacobi paramétrée par les données ?

Un des objectifs de la Semaine d'Etude Maths-Entreprises se voulait d'amorcer le début d'une réflexion collective de jeunes chercheurs autour d'un sujet proposé par une entreprise, en l'occurrence le consortium pétrolier et universitaire GOCAD. Il s'agit ici de pouvoir reconstruire le sous-sol rapidement en ayant seulement la connaissance d'un certain nombre de données discrètes éparses. Si nous avons précédemment présenté les différentes approches auxquelles nous avons eu accès, nous allons maintenant tenter de préciser les idées qui ont pu émerger au cours de cette semaine d'étude.

Bien que l'approche statistique décrite dans [2] soit très performante lorsqu'il s'agit de reconstruire le sous-sol à partir de points, elle fait en revanche appel à un modèle sous-jacent de variabilité, ce qui la limite en nombre et type de données assimilables. C'est pourquoi la démarche adoptée dans [3] incorpore directement toutes les informations disponibles dans le système discrétisé puis résout numériquement le problème en un temps raisonnable. Cependant, la courbure de certaines couches obtenues par la méthode précédente ne semble pas géologiquement acceptable.

C'est aussi une des raisons pour lesquelles nous nous sommes penchés sur l'approche développée dans [1] car elle fournit un modèle théorique simple qui mélange processus de dépôts et de déformations mécaniques. Une telle démarche permet-elle une reconstitution globale du sous-sol à partir de certaines données imposées ? En utilisant bien ces dernières, il s'agit de judicieusement paramétrer la loi d'évolution grâce à l'expertise du géologue, puis de résoudre numériquement de manière efficace les équations de Hamilton-Jacobi associées grâce au savoir-faire de l'ingénieur.

2.1 Présentation du modèle retenu

On s'inspire ici surtout de l'approche développée dans [1]. On rappelle s'être placé dans le cadre des méthodes *level-set*. Autrement dit, la couche géologique associée à l'instant t est implicitement représentée par une certaine surface notée Γ_t. Cela signifie qu'elle n'est pas explicitement caractérisée par l'énumération des points $\mathbf{x}$ de l'espace par lesquels elle passe, mais plutôt par les points d'annulation d'une certaine fonction $\Psi(t,.)$ appelée champ scalaire et définie sur l'espace usuel $\mathbb{R}^3$ tout entier. En termes mathématiques, on écrit :

$$\Gamma_t = \{\mathbf{x} \in \mathbb{R}^3, \quad \Psi(t,\mathbf{x}) = 0\}.$$

Afin de pouvoir relier les différentes couches Γ_t entre elles, on suppose qu'il existe une loi d'évolution notée H dont un choix adéquat fera l'objet de la prochaine sous-section. Plus précisément, elle permet de relier n'importe quel couple de champs $\Psi(t,.)$ et $\Psi(\tilde{t},.)$ entre eux. Elle dépend évidemment de l'instant t et du point de l'espace $\mathbf{x}$ considérés, mais aussi de l'orientation prise par le flot Ψ en cet endroit et à ce moment là, qui est caractérisée par le vecteur unitaire suivant :

$$\vec{\mathbf{n}}(t,\mathbf{x}) = \frac{\vec{\nabla}\Psi(t,\mathbf{x})}{\|\vec{\nabla}\Psi(t,\mathbf{x})\|}.$$

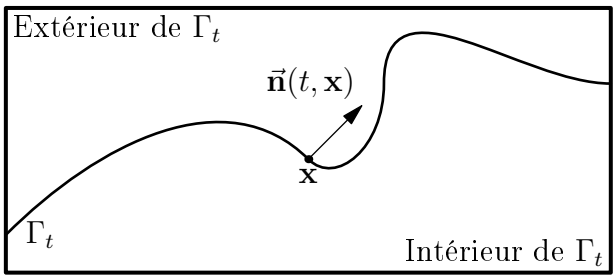

FIGURE 10 – *Le domaine d'étude est assimilé à une boîte que la surface Γ_t sépare en deux parties. La partie supérieure (resp. inférieure) correspond à l'extérieur (resp. l'intérieur) de Γ_t puisqu'elle contient les points appartenant aux couches plus récentes (resp. âgées).*

Pour les points $\mathbf{x}$ appartenant à Γ_t, le vecteur $\vec{\mathbf{n}}(t,\mathbf{x})$ représente simplement la normale unitaire en $\mathbf{x}$ à la surface Γ_t, orientée de l'intérieur vers l'extérieur de Γ_t (cf. Figure 10). L'évolution de la forme des surfaces Γ_t et plus globalement des champs $\Psi(t,.)$ au cours du temps est alors régie par une équation aux dérivées partielles dite de Hamilton-Jacobi. Modélisant un phénomène de propagation, elle est de type hyperbolique, comme l'équation des ondes, et s'écrit sous la forme suivante :

$$\frac{\partial \Psi}{\partial t}(t,\mathbf{x}) + H\left(\mathbf{x}, \vec{\nabla}\Psi\,(t,\mathbf{x})\right) = 0.$$

On pourrait également faire dépendre la loi H des propriétés d'ordre deux du flot Ψ, par exemple de la quantité :

$$\kappa(t,\mathbf{x}) = \text{div}\ \vec{\mathbf{n}}(t,\mathbf{x}) = \vec{\nabla}.\left(\frac{\vec{\nabla}\Psi(t,\mathbf{x})}{\|\vec{\nabla}\Psi(t,\mathbf{x})\|}\right).$$

Pour les points $\mathbf{x}$ appartenant à Γ_t, le nombre $\kappa(t,\mathbf{x})$ représente simplement la courbure moyenne en $\mathbf{x}$ de la surface Γ_t. Cependant, cette contribution parabolique, malgré son effet régularisant du type équation de la chaleur, compliquerait la résolution numérique et alourdirait le temps de calcul (condition CFL plus contraignante). Par souci de simplicité et surtout par manque de temps, nous avons décidé de ne pas faire dépendre H de κ.

2.2 Description de la loi d'évolution

Tout d'abord, on commence par discrétiser la partie du sous-sol terrestre qu'on souhaite reconstruire de telle sorte que les failles qu'elle contient se retrouvent forcément sur les bords de certains tétraèdres $\mathcal{T}$. Celles-ci seront alors numériquement considérées comme le bord du domaine d'étude, comme cela est fait dans [3] en dédoublant les valeurs de Ψ sur les sommets des $\mathcal{T}$. Ainsi, une strate géologique traversée par une faille sera assimilée à deux couches évoluant indépendamment de part et d'autre de la faille en question.

Ensuite, on suppose qu'il y a exactement I orientations connues à incorporer dans la loi d'évolution H. Elles sont représentées par des vecteurs unitaires notés $\vec{\mathbf{a}}_i$ qui sont respectivement localisés dans le domaine aux points $\mathbf{x}_i$, pour tout entier i allant de 1 à I. En s'inspirant de la première des expressions page 756 donnée dans [1] Section 5, on fait l'hypothèse que la loi d'évolution peut se mettre sous la forme qui suit, les λ_i (resp. μ) paramétrant l'adéquation du modèle par rapport aux $\vec{\mathbf{a}}_i$ (resp. aux données ponctuelles) :

$$H\left(\mathbf{x}, \vec{\nabla}\Psi\left(t,\mathbf{x}\right)\right) = \mu\left(\mathbf{x}\right)\left(\lambda_0\left(\mathbf{x}\right)\|\vec{\nabla}\Psi\left(t,\mathbf{x}\right)\| + \sum_{i=1}^{I}\lambda_i\left(\mathbf{x}\right)\vec{\mathbf{a}}_i.\vec{\nabla}\Psi\left(t,\mathbf{x}\right)\right).$$

Finalement, on suppose connaître à certains endroits l'emplacement de K horizons géologiques différents, qu'on a préalablement ordonnés du plus âgé au plus récent. Pour tout entier k allant de 1 à K, on fait l'hypothèse que dans la strate k, il y a J_k points distincts repérés aux coordonnées $\mathbf{y}_j^k$ pour j allant de 1 à J_k. Le calibrage du modèle sur ces points sera géré par le paramètre μ. La Figure 11 résume les notations introduites et on va maintenant décrire plus en détail les choix effectués pour l'expression des λ_i et μ.

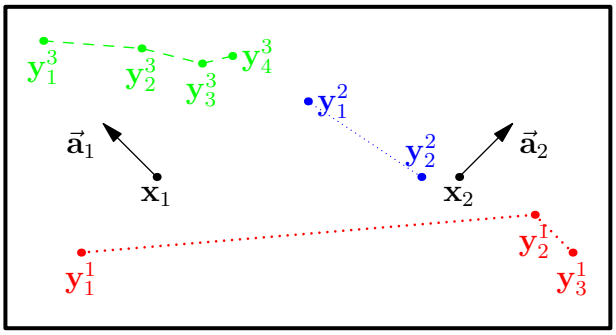

FIGURE 11 – *Illustration des notations introduites pour les données connues dans le cas de deux vecteurs ainsi que de neuf points appartenant à trois horizons géologiques différents.*

2.3 Expression des différents paramètres associés à la loi d'évolution

On rappelle que les λ_i régissent l'influence des vecteurs tandis que le paramètre μ va prendre en compte celle des points. Tout d'abord, lorsqu'on cherche l'équation des caractéristiques associée à la loi H dans le cas particulier où Ψ est de la forme $T(\mathbf{x}) - t$, comme cela est fait dans [1], on obtient en prenant $\mu = 1$:

$$\dot{\mathbf{x}} = \lambda_0\vec{\mathbf{n}} + \sum_{i=1}^{I}\lambda_i\vec{\mathbf{a}}_i.$$

Ainsi, dans ce cas là, un point $\mathbf{x}$ appartenant à Γ_t aura tendance à être localement déplacé dans la direction formée par une combinaison linéaire des différents vecteurs $\vec{\mathbf{a}}_i$ et de la normale $\vec{\mathbf{n}}$, dont les coefficients sont justement les λ_i qu'on veut définir.

En première approche, on fait donc l'hypothèse qu'en dehors d'un disque de centre $\mathbf{x}_i$, i.e. le point où $\vec{\mathbf{a}}_i$ se situe, et de rayon r_i à calibrer, le vecteur $\vec{\mathbf{a}}_i$ n'influera pas du tout. De plus, par souci de simplicité, on suppose que les $\vec{\mathbf{a}}_i$ ne sont pas trop proches les uns des autres afin d'imposer aux I disques de ne pas s'intersecter entre eux sans pour autant devoir choisir des rayons d'influence r_i ridiculement petits. Plus précisément, cela se traduit par la condition suivante :

$$\max_{i\in\{1,\dots,I\}} r_i \;<\; \min_{(i,j)\in\{1,\dots,I\}^2} \|\mathbf{x}_i - \mathbf{x}_j\|.$$

Par conséquent, ayant pris soin de disjoindre les disques d'influence entre eux, seules deux situations peuvent se produire : ou bien le point $\mathbf{x}$ de Γ_t considéré ne se situe dans aucun des I disques, auquel cas on impose à la caractéristique de bouger suivant la normale, c'est-à-dire que $\lambda_0(\mathbf{x}) = 1$ et $\lambda_i(\mathbf{x}) = 0$ pour tout entier $i \in \{1, \dots, I\}$; ou bien $\mathbf{x}$ appartient à l'un d'eux seulement, mettons le disque i_0, et on force alors la caractéristique à suivre la direction $\vec{\mathbf{a}}_{i_0}$ selon un poids $\lambda_{i_0}(\mathbf{x}) \in [0, 1]$ dont la dépendance en $\mathbf{x}$ sera une fonction régulière décroissante de la distance entre $\mathbf{x}$ et $\mathbf{x}_{i_0}$, le reste étant alloué à la normale en fixant $\lambda_0(\mathbf{x}) = 1 - \lambda_{i_0}(\mathbf{x})$ et $\lambda_i(\mathbf{x}) = 0$ pour tout entier $i \neq i_0$.

De manière plus générale, l'idée sous-jacente au raisonnement précédent consiste à imposer aux coefficients λ_i de former une partition de l'unité, qui est en quelque sorte une généralisation spatiale du concept de combinaison linéaire convexe. Les coefficients doivent être des fonctions régulières positives dont la somme vaut tout le temps un et dont le support est celui du disque d'influence associé. Dans notre cas, pour tout point $\mathbf{x}$ de Γ_t, on a posé $\lambda_0(\mathbf{x}) = 1 - \sum_{i=1}^{I} \lambda_i(\mathbf{x})$ ainsi que (cf. Figure 12) :

$$\forall i \in \{1, \dots, I\}, \quad \lambda_i(\mathbf{x}) = \begin{cases} e^{\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\|\mathbf{x}-\mathbf{x}_i\|^2 - r_i^2}} & \text{si} \quad \|\mathbf{x} - \mathbf{x}_i\| < r_i \\ 0 & \text{sinon.} \end{cases}$$

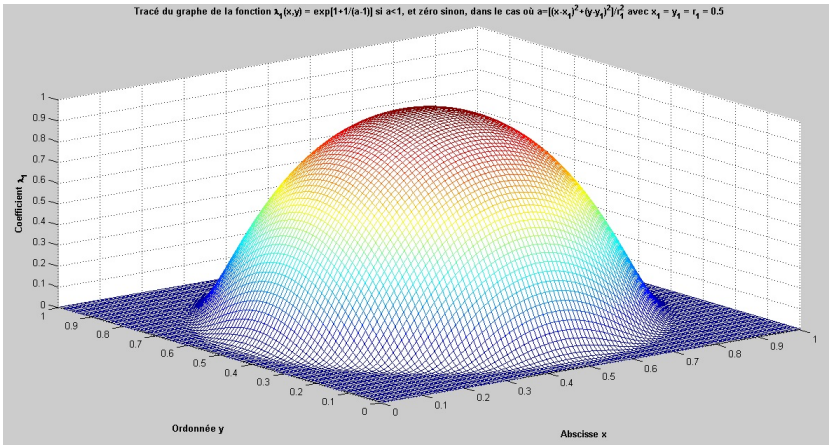

FIGURE 12 – *Tracé du graphe d'un coefficient λ_1 dans le cas 2-D où un vecteur $\vec{\mathbf{a}}_1$ est imposé au point $(\frac{1}{2}, \frac{1}{2})$ avec un rayon d'influence $r_1 = \frac{1}{2}$.*

Quant au paramètre μ, il suffit de regarder attentivement l'équation de Hamilton-Jacobi pour s'apercevoir qu'il agira sur l'ordre de grandeur avec lequel le champ $\Psi(t,.)$ sera localement modifié par la loi H au cours du temps, alors que les λ_i influençaient la direction par laquelle cette déformation avait lieu. En effet, plus il devient nul et plus $\Phi(t,.)$ reste constant au cours du temps et donc moins la surface Γ_t évolue. Au contraire, celui-ci n'aura aucune influence s'il est proche de un.

C'est donc en jouant spatialement sur cette propriété de μ que la surface Γ_t va être contrainte de se rapprocher plus ou moins vite des points $\mathbf{y}_j^k$ imposés. Au sein des K couches géologiques connues, on sélectionne celles qui sont plus récentes que Γ_t et parmi ces dernières, on note k_0 la plus âgée, c'est-à-dire celle qui se rapproche le plus de Γ_t en terme d'âge. S'il n'en existe pas, on pose tout simplement $\mu = 1$. Sinon on considère un point $\mathbf{x}$ de Γ_t ainsi que les projetés orthogonaux $p_{\Gamma_t}(\mathbf{y}_j^{k_0})$ des points $\mathbf{y}_j^{k_0}$ sur la surface Γ_t :

$$\forall j \in \{1, \dots, J_{k_0}\}, \quad p_{\Gamma_t}\left(\mathbf{y}_j^{k_0}\right) = \min_{\mathbf{y} \in \Gamma_t} \|\mathbf{y} - \mathbf{y}_j^{k_0}\|.$$

On recherche alors l'indice $j_0 \in \{1, \dots, J_{k_0}\}$ pour lequel la distance géodésique sur Γ_t entre le point $\mathbf{x}$ et le projeté $p_{\Gamma_t}(\mathbf{y}_{j_0}^{k_0})$ est la plus faible. Pour cela, on considère pour chaque entier $j \in \{1, \dots, J_{k_0}\}$, parmi toutes les courbes paramétrées $\gamma : s \in [0,1] \mapsto \gamma(s) \in \Gamma_t$ telle que $\gamma(0) = \mathbf{x}$ et $\gamma(1) = p_{\Gamma_t}(\mathbf{y}_j^{k_0})$, celle notée γ_j dont la longueur $l(\gamma) = \int_0^1 \|\dot{\gamma}(s)\| ds$ est la plus petite, et on appelle j_0 l'indice pour lequel $l(\gamma_{j_0}) = \min_{j \in \{1,\dots,J_{k_0}\}} l(\gamma_j)$. Puis, on pose (cf. Figure 13) :

$$\mu(\mathbf{x}) = \frac{\|p_{\Gamma_t}\left(\mathbf{y}_{j_0}^{k_0}\right) - \mathbf{y}_{j_0}^{k_0}\|}{1 + \|p_{\Gamma_t}\left(\mathbf{y}_{j_0}^{k_0}\right) - \mathbf{y}_{j_0}^{k_0}\|}.$$

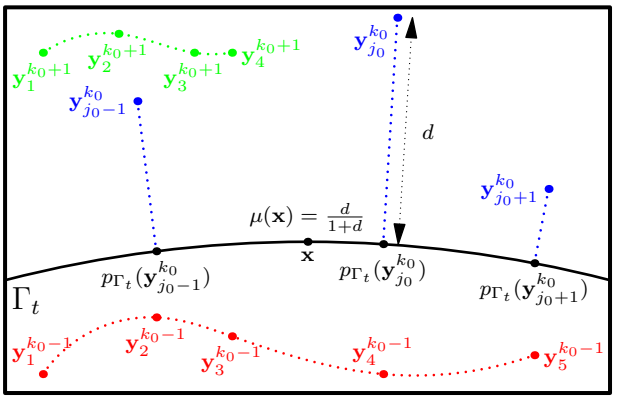

FIGURE 13 – *Illustration du calcul de μ dans un cadre bidimensionnel.*

Par conséquent, plus le point $\mathbf{y}_{j_0}^{k_0}$ est éloigné de son projeté, plus les points $\mathbf{x}$ de Γ_t qui l'entourent se déplaceront suivant la loi H. Au contraire, au fur et à mesure que le temps passe, la surface Γ_t va se positionner près des points de l'horizon k_0 et ne plus évoluer puisqu'alors μ deviendra quasiment nul. Quand le régime stationnaire est atteint, on réinitialise le calcul de μ par rapport à l'horizon $k_0 + 1$, et si $k_0 = K$, on pose $\mu = 1$.

3 Résolution numérique des équations de Hamilton-Jacobi

Au cours de cette Semaine d'Etude Maths-Entreprises, le consortium pétrolier GOCAD nous a proposé de réfléchir sur les différentes manières de reconstruire le sous-sol terrestre à partir de données discrètes éparses. Après avoir pris connaissance d'un certain nombre d'articles, notamment de l'approche statistique effectuée dans [2] ainsi que de la démarche géologique et numérique développée dans [3], nous nous sommes attardés sur le point de vue théorique adopté dans [1] et avons tenté de l'adapter à notre problématique.

Ayant désormais un nouveau modèle à notre disposition, il s'agit maintenant de le résoudre numériquement et c'est l'objet de cette section. On rappelle que chaque couche géologique est une surface de la forme $\Gamma_t = \{\mathbf{x} \in \mathbb{R}^3, \quad \Psi(t, \mathbf{x}) = 0\}$ et que son évolution au sein du sous-sol est gérée par l'équation de Hamilton-Jacobi suivante, les poids λ_i (respectivement μ) paramétrant l'adéquation du modèle par rapport aux orientations $\vec{\mathbf{a}}_i$ connues (respectivement par rapport aux points imposés) :

$$\frac{\partial \Psi}{\partial t}(t, \mathbf{x}) + \mu(\mathbf{x}) \left(\lambda_0(\mathbf{x}) \|\vec{\nabla}\Psi(t, \mathbf{x})\| + \sum_{i=1}^{I} \lambda_i(\mathbf{x}) \vec{\mathbf{a}}_i . \vec{\nabla}\Psi(t, \mathbf{x}) \right) = 0.$$

Par souci de simplicité, on suppose ici que le domaine d'étude ne contient pas de failles, même si l'article [3] explique comment les prendre en compte dans la discrétisation. De plus, on se place dans un cadre bidimensionnel où la variable spatiale est donc $\mathbf{x} = (x, y) \in \mathbb{R}^2$. Afin que le problème soit bien posé, il est aussi nécessaire de spécifier une surface initiale $\Gamma_1 = \{(x, y) \in \mathbb{R}^2, \quad \Psi(0, x, y) = 0\}$. Dans notre cas, quitte à se placer suffisamment en profondeur, à l'altitude y_0 par exemple, on supposera que Γ_1 est une droite c'est-à-dire :

$$\forall (x, y) \in \mathbb{R}^2, \quad \Psi(0, x, y) = y - y_0.$$

Deux approches ont alors été considérées. La première consiste à discrétiser les points de la surface Γ_1 puis de les faire glisser au fur et à mesure dans la direction suggérée par l'équation des caractéristiques. Au contraire, la seconde considère l'ensemble du domaine d'étude discrétisé puis cherche à faire évoluer la valeur du flot Ψ en chaque point de la grille grâce à l'équation de Hamilton-Jacobi, la surface Γ_t étant ensuite déterminée à l'aide d'un algorithme de recherche de contour.

3.1 Une approche lagrangienne

Comme cela est fait dans [1], on suppose ici que le flot Ψ peut se mettre sous la forme particulière suivante : $\Psi(t, x, y) = \varphi(x, y) - t$ où la fonction φ est celle avec laquelle on va préférer travailler. En effet, il vient alors $\Gamma_t = \{(x, y) \in \mathbb{R}^2, \quad \varphi(x, y) = t\}$ mais aussi $\frac{\partial \Psi}{\partial t}(t, x, y) = -1$ ainsi que $\vec{\nabla}\Psi(t, x, y) = \vec{\nabla}\varphi(x, y)$. Il en résulte que l'équation de Hamilton-Jacobi ayant pour inconnue le flot Ψ se transforme en une équation eikonale indépendante du temps qui gère la répartition spatiale de la fonction φ :

$$H\left(\mathbf{x}, \vec{\nabla}\varphi(\mathbf{x})\right) = \mu(x, y) \left(\lambda_0(x, y) \|\vec{\nabla}\varphi(x, y)\| + \sum_{i=1}^{I} \lambda_i(x, y) \vec{\mathbf{a}}_i . \vec{\nabla}\varphi(x, y) \right) = 1.$$

Ainsi, pour retrouver Γ_t à partir de φ, il suffit de calculer les lignes de niveaux de φ, à l'aide par exemple d'un algorithme de recherche de contour. Par ailleurs, on introduit la notation $\vec{\mathbf{p}} = \vec{\nabla}\varphi(\mathbf{x})$. Ce sera la seconde variable de H et elle sera supposée indépendante de la première, à savoir $\mathbf{x}$. On cherche alors l'équation des caractéristiques associées à l'équation de transport précédente $H(\mathbf{x}, \vec{\mathbf{p}}) = 1$. Il s'agit des courbes $s \mapsto \mathbf{x}(s)$ et $s \mapsto \vec{\mathbf{p}}(s)$ sur lesquelles la fonction $s \mapsto H(\mathbf{x}(s), \vec{\mathbf{p}}(s))$ est constante. Or, on a :

$$\dot{H}(s) = \dot{\mathbf{x}}(s).\vec{\nabla}_{\mathbf{x}} H(\mathbf{x}(s), \vec{\mathbf{p}}(s)) + \dot{\vec{\mathbf{p}}}(s).\vec{\nabla}_{\vec{\mathbf{p}}} H(\mathbf{x}(s), \vec{\mathbf{p}}(s))$$

donc en prenant en particulier $\dot{\mathbf{x}} = \vec{\nabla}_{\vec{\mathbf{p}}} H$ ainsi que $\dot{\vec{\mathbf{p}}} = -\vec{\nabla}_{\mathbf{x}} H$, on obtient bien $\dot{H} = 0$. On transforme ainsi l'équation aux dérivées partielles $H(\mathbf{x}, \vec{\mathbf{p}}) = 1$ en un système couplé d'équations différentielles ordinaires vérifiées sur les caractéristiques, à savoir :

$$\begin{cases} \dot{\vec{\mathbf{p}}} = -\vec{\nabla}\mu(\mathbf{x}) \left(\lambda_0(\mathbf{x}) \|\vec{\mathbf{p}}\| + \sum_{i=1}^{I} \lambda_i(\mathbf{x}) \vec{\mathbf{a}}_i.\vec{\mathbf{p}} \right) - \mu(\mathbf{x}) \left(\vec{\nabla}\lambda_0(\mathbf{x}) \|\vec{\mathbf{p}}\| + \sum_{i=1}^{I} \vec{\nabla}\lambda_i(\mathbf{x}) \left(\vec{\mathbf{a}}_i.\vec{\mathbf{p}}\right) \right) \\ \dot{\mathbf{x}} = \mu(\mathbf{x}) \left(\lambda_0(\mathbf{x}) \dfrac{\vec{\mathbf{p}}}{\|\vec{\mathbf{p}}\|} + \sum_{i=1}^{I} \lambda_i(\mathbf{x}) \vec{\mathbf{a}}_i \right). \end{cases}$$

Ensuite, on considère un pas de temps $\Delta t > 0$ suffisamment petit. L'intervalle $[0, T]$ sur lequel on veut simuler l'évolution du système précédent est alors discrétisé en certains instants $\{t_1, \ldots, t_N\}$ où $t_n = (n-1)\Delta t$ pour tout entier $n \in \{1, \ldots, N\}$ avec $N = 1 + \frac{T}{\Delta t}$. On suppose aussi connaître les positions et orientations de Γ_{t_n} au sein du domaine d'étude, mettons en L points très rapprochés les uns des autres, respectivement notées $\mathbf{x}_l^n$ et $\vec{\mathbf{p}}_l^n$ pour tout $l \in \{1, \ldots, L\}$. Un schéma d'Euler explicite est alors utilisé pour la discrétisation :

$$\begin{cases} \vec{\mathbf{p}}_l^{n+1} = \vec{\mathbf{p}}_l^n - \Delta t \vec{\nabla}\mu_l^n \left(\lambda_{0,l}^n \|\vec{\mathbf{p}}_l^n\| + \sum_{i=1}^{I} \lambda_{i,l}^n (\vec{\mathbf{a}}_i.\vec{\mathbf{p}}_l^n) \right) \\ \qquad\qquad - \mu_l^n \Delta t \left(\vec{\nabla}\lambda_{0,l}^n \|\vec{\mathbf{p}}_l^n\| + \sum_{i=1}^{I} \vec{\nabla}\lambda_{i,l}^n \left(\vec{\mathbf{a}}_i.\vec{\mathbf{p}}_l^n\right) \right) \\ \mathbf{x}_l^{n+1} = \mathbf{x}_l^n + \mu_l^n \Delta t \left(\lambda_{0,l}^n \dfrac{\vec{\mathbf{p}}_l^n}{\|\vec{\mathbf{p}}_l^n\|} + \sum_{i=1}^{I} \lambda_{i,l}^n \vec{\mathbf{a}}_i \right) \end{cases}$$

où $\lambda_{i,l}^n = \lambda_i(\mathbf{x}_l^n)$, $\mu_l^n = \mu(\mathbf{x}_l^n)$, $\vec{\nabla}\lambda_{i,l}^n = \vec{\nabla}\lambda_i(\mathbf{x}_l^n)$ et $\vec{\nabla}\mu_l^n = \vec{\nabla}\mu(\mathbf{x}_l^n) \approx \frac{(\mu_{l+1}^n - \mu_l^n)(\mathbf{x}_{l+1}^n - \mathbf{x}_l^n)}{\|\mathbf{x}_{l+1}^n - \mathbf{x}_l^n\|}$.

Finalement, les Figures 14 et 15 donnent un aperçu des résultats qu'on peut obtenir grâce à cet algorithme. Elles illustrent respectivement la reconstruction 2-D d'un sous-sol sans failles à partir d'une donnée directionnelle et de mesures ponctuelles provenant de deux horizons géologiques différents. En particulier, on observe très clairement sur la Figure 15 le phénomène stationnaire mentionné qui apparaît quand la strate vient se positionner près des points connus, et qui est dû au comportement spatial qu'on a choisi pour μ.

De plus, on identifie des discontinuités nettes de la normale le long de la surface Γ_t. Celles-ci proviennent du fait que μ est constant par morceaux. En effet, chacun des points imposés agit à travers une des différentes valeurs de μ sur un certain morceau de Γ_t. Chaque portion de courbes va donc évoluer de manière distincte mais comme Γ_t est toujours continue, des coins apparaissent aux endroits où le comportement diffère. On expliquera plus loin comment choisir un μ qui puisse assurer la continuité de la normale sur Γ_t.

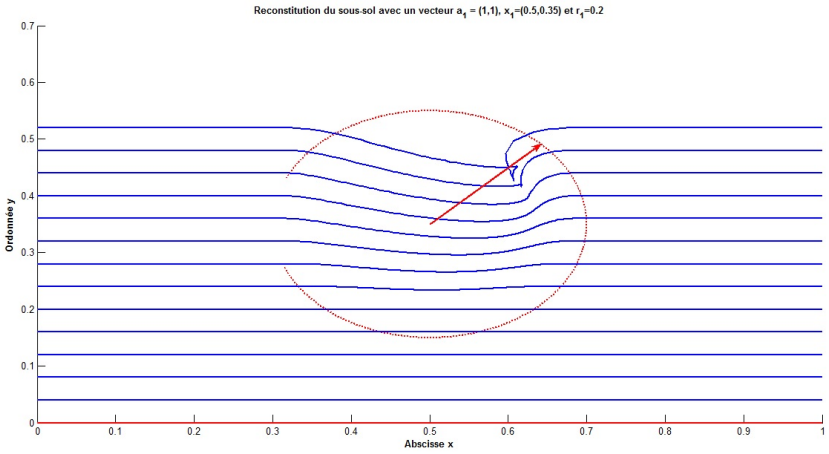

FIGURE 14 – *Reconstitution du sous-sol géologique à partir d'un vecteur.*

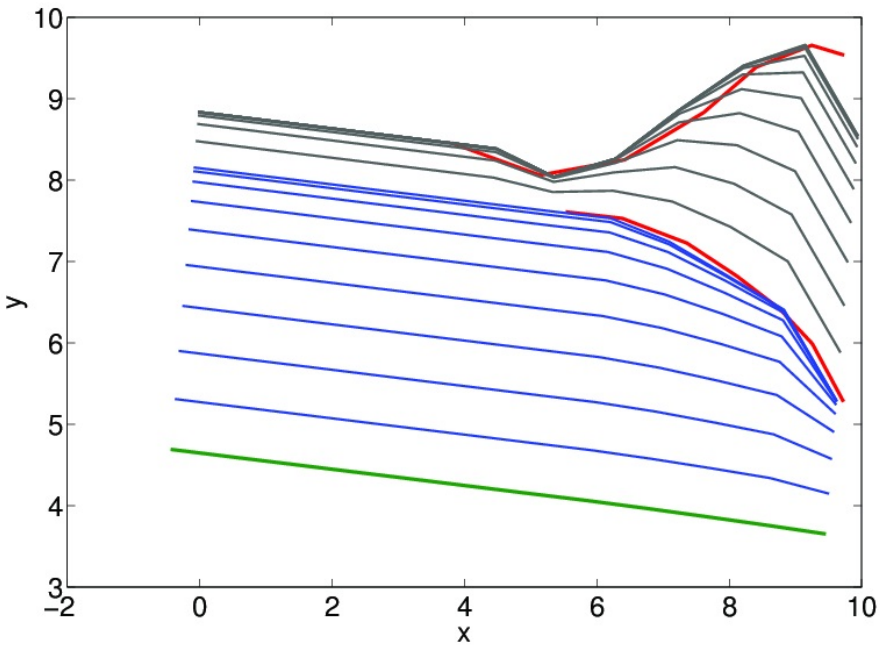

FIGURE 15 – *Reconstitution du sous-sol géologique à partir de points.*

Quant à l'influence des vecteurs, elle est très mal prise en compte par l'algorithme, comme on peut le voir sur la Figure 14. La déformation est seulement visible à la sortie du disque d'influence avec une orientation différente de celle qui est imposée. On souhaiterait plutôt qu'elle soit effective dans la bonne direction et dès que la surface atteint le point où elle se situe. De plus, elle est locale dans le sens où elle n'agit pas sur la direction que prend globalement la surface. On détaillera après comment faire pour pallier à cela.

Bien que l'approche décrite ci-dessus soit assez naïve, elle reste avant tout très simple et très rapide à mettre en œuvre numériquement avec un temps de calcul très raisonnable. De plus, elle semble très bien adaptée au cas 2-D sans failles où des points sont imposés. En revanche, elle prend mal en compte les vecteurs et de nombreux problèmes surviennent quand la répartition des contraintes ponctuelles et directionnelles devient plus complexe, ce qui la rend difficilement généralisable à un cadre tridimensionnel.

Plus précisément, un phénomène d'accumulation des points de la surface est observé dans certains cas. Ceci est simplement lié au fait qu'on effectue un glissement des points et que si la loi d'évolution considérée décide de les concentrer en une zone particulière, alors ils s'y rassemblent. Pire encore, il arrive souvent que ceux-ci se croisent comme sur la Figure 14, ou qu'ils sortent du domaine d'étude. En 2-D, le problème peut se régler en supprimant/ajoutant des points de/sur la ligne qui sont trop proches/éloignés.

Cependant, la chirurgie qui est effectuée sur la topologie d'une courbe est difficilement généralisable au cas d'une surface dont la géométrie peut devenir extrêmement complexe et suffisamment tordue pour qu'un algorithme simple ne puisse la réparer convenablement. Pour finir, il semble assez difficile de prendre en compte correctement les failles avec cette méthode, ce qui l'empêche d'être appliquée à des situations réalistes de reconstitution de sous-sol géologique.

C'est pourquoi l'approche qu'on va maintenant détailler consiste non pas à bouger directement les points de Γ_t mais plutôt à considérer l'évolution du flot Ψ dont une ligne de niveaux représente Γ_t implicitement. Ceci permettra de généraliser quasi-immédiatement la 2-D à la 3-D et de bien gérer les changements de topologie sur Γ_t (apparition de vide, auto-intersections, ...). Ce sont d'ailleurs les deux avantages des méthodes *level-set* qui expliquent en partie leur récent succès.

3.2 Une approche eulérienne

La démarche qu'on va ici adopter semble être celle qu'il faille suivre pour obtenir des résultats numériques pertinents. Elle utilise les techniques décrites dans [4] et elle a été développée par J. Dalphin à la suite de la Semaine d'Etude Maths-Entreprises. En effet, comme on a pu le voir, la simulation d'équations de type Hamilton-Jacobi peut s'avérer délicate, même pour une loi simple, car il s'agit de phénomènes de transport.

Dans l'approche eulérienne, les failles peuvent facilement être prises en compte grâce à l'astuce mentionnée dans [3]. En effet, si la discrétisation du domaine d'étude est réalisée de telle sorte qu'elles se retrouvent sur le bord de certains tétraèdres, alors il suffit de dédoubler les valeurs de Ψ aux sommets des faces les contenant, mais nous ne nous attarderons plus là-dessus en supposant que le domaine considéré n'en contient pas.

De plus, comme cela est très clairement expliqué dans [4], la méthode qui va suivre ne dépend quasiment pas de la dimension de l'espace dans lequel on travaille. Ainsi, passer de la 2-D à la 3-D ne pose a priori aucun problème, ce qui n'était pas le cas de l'approche lagrangienne précédente. On se place donc dans un cadre bidimensionnel où la variable spatiale est notée $\mathbf{x} = (x, y) \in \mathbb{R}^2$.

3.2.1 Première étape : discrétisation du domaine spatial et temporel

Tout d'abord, le domaine d'étude est discrétisé en une grille fixe formée de PQ points notés $\mathbf{x}_{p,q} = (x_p, y_q)$ pour tout entier $p \in \{1, \dots, P\}$ et $q \in \{1, \dots, Q\}$. Puis, après avoir sélectionné un pas de temps $\Delta t > 0$ suffisamment petit, l'intervalle de temps $[0, T]$ sur lequel on veut simuler l'évolution du flot Ψ est assimilé à N instants différents $\{t_1, \dots, t_N\}$ où $t_n = (n-1)\Delta t$ pour tout entier $n \in \{1, \dots, N\}$ avec $N = 1 + \frac{T}{\Delta t}$.

Soit un entier $n \in \{1, \dots, N-1\}$ désormais fixé. On suppose connaître à l'instant t_n la valeur du flot Ψ en chacun des points $\mathbf{x}_{p,q}$ du maillage qu'on note $\Psi^n_{p,q} = \Psi(t_n, x_p, y_q)$. On cherche à calculer $\Psi^{n+1}_{p,q}$ pour tout entier $p \in \{1, \dots, P\}$ et $q \in \{1, \dots, Q\}$. On explique maintenant comment procéder mais on va auparavant faire une hypothèse supplémentaire sur la structure du flot Ψ : le fait d'être une distance signée.

3.2.2 Deuxième étape : (re-)initialiser le champ scalaire en distance signée

De manière assez générale, lorsqu'il représente implicitement une surface Γ_t, un champ scalaire $\Psi(t, .)$ est qualifié de distance signée s'il vérifie les trois propriétés suivantes :
- il est négatif à l'intérieur de Γ_t et positif à l'extérieur de Γ_t (cf. Figure 10) ;
- il s'annule sur la surface c'est-à-dire $\Gamma_t = \{(x, y) \in \mathbb{R}^2, \quad \Psi(t, x, y) = 0\}$;
- la norme de son gradient vaut toujours l'unité i.e. $\forall \mathbf{x} \in \mathbb{R}^2, \quad \|\vec{\nabla}\Psi(t, \mathbf{x})\| = 1$.

En particulier, le champ $\Psi(0, x, y) = y - y_0$ qui est associé à la surface initiale $\Gamma_1 = \Gamma_{t_1}$ satisfait bien ces trois critères. D'après [4], la notion de distance signée est fondamentale lorsqu'on simule la dynamique de surfaces implicites car elle permet au flot de ne pas se dégrader au cours du temps et assure ainsi un comportement numérique efficace de la solution à long terme.

On fait donc l'hypothèse que le champ $\Psi^n = \Psi(t_n, .)$ est également une distance signée. Si ce n'est pas le cas, alors on recherche la surface $\Gamma_n = \Gamma_{t_n} = \{(x, y) \in \mathbb{R}^2, \ \Psi(t_n, x, y) = 0\}$ à l'aide d'un algorithme de contour. On obtient ainsi L points de Γ_n notés $(\mathbf{x}^n_l)_{1 \leqslant l \leqslant L}$ qui ne correspondent pas forcément à des nœuds $\mathbf{x}_{p,q}$ de la grille. On pose alors pour tout entier $p \in \{1, \dots, P\}$ et $q \in \{1, \dots, Q\}$:

$$\Psi^n_{p,q} = \Psi(t_n, x_p, y_q) = \varepsilon \, d\left(\mathbf{x}_{p,q}, \Gamma_n\right) = \varepsilon \, \left\| p_{\Gamma_n}\left(\mathbf{x}_{p,q}\right) - \mathbf{x}_{p,q} \right\| \approx \varepsilon \min_{l \in \{1, \dots, L\}} \left\| \mathbf{x}_{p,q} - \mathbf{x}^n_l \right\|$$

où p_{Γ_n} désigne le projeteur orthogonal sur Γ_n avec $\varepsilon = -1$ (respectivement $\varepsilon = 1$) si le point $\mathbf{x}_{p,q}$ est à l'intérieur (respectivement à l'extérieur) de Γ_n. Le champ Ψ^n obtenu vérifie bien les trois propriétés de la distance signée car $\mathbf{x} \mapsto d(\mathbf{x}, \Gamma_n)$ est différentiable presque partout avec un gradient unitaire. Une démarche similaire sera effectuée lorsqu'on aura calculé la valeur de Ψ^{n+1} sur Γ_{n+1} et qu'il faudra l'estimer en tous les points de la grille.

3.2.3 Troisième étape : calcul du paramètre μ et des coefficients λ_i

La méthode précédente avait fait apparaître des discontinuités de la normale le long de la surface Γ_t dues au choix d'un μ constant par morceaux. On décrit maintenant comment pallier ce problème. Rappelons qu'on dispose de J_k données ponctuelles notées $(\mathbf{y}_j^k)_{1\leqslant j\leqslant J_k}$ sur le k-ième horizon pour tout entier $k \in \{1, ..., K\}$, les K horizons ayant été préalablement ordonnés du plus vieux au plus jeune. Parmi ces derniers, on considère ceux qui sont plus récents que $\Gamma_n = \Gamma_{t_n}$ et le plus âgé d'entre eux est noté k_n. S'il n'en existe aucun, on pose tout simplement $\mu = 1$.

Dans le cas contraire, pour chaque entier $j \in \{1, \ldots, J_{k_n}\}$, on calcule alors le projeté orthogonal $p_{\Gamma_n}(\mathbf{y}_j^{k_n})$ du point $\mathbf{y}_j^{k_n}$ sur la surface Γ_n : il s'agit de l'un des L points $(\mathbf{x}_l^n)_{1\leqslant l\leqslant L}$ fournis par l'algorithme de recherche de contour et l'indice lui correspondant est noté l_j. S'il y en a deux, ce qui peut arriver, alors on considère l'indice le plus faible. En général, les points approximant Γ_n sont ordonnés de façon à ce que la courbe Γ_n soit parcourue par les $\mathbf{x}_l^n$ de gauche à droite quand l'entier l varie de 1 à L, et de même pour les $(\mathbf{x}_{l_j}^n)_{1\leqslant j\leqslant J_{k_n}}$ quand j va de 1 à J_{k_n}, ce qu'on va dorénavant supposer. Avec les notations, on a donc :

$$l_j = \min\left\{ l \in \{1, \ldots, L\},\ \ \|\mathbf{x}_l^n - \mathbf{y}_j^{k_n}\| = d\left(\mathbf{y}_j^{k_n}, \Gamma_n\right) = \min_{1\leqslant \tilde{l}\leqslant L} \|\mathbf{x}_{\tilde{l}}^n - \mathbf{y}_j^{k_n}\| \right\}.$$

Soit un entier $l \in \{1, \ldots, L\}$ désormais fixé. On rappelle qu'on veut ici définir $\mu(\mathbf{x}_l^n)$ dont la valeur retenue va être une combinaison linéaire des précédents μ. Mais pour cela, il nous faut auparavant introduire pour chacun des entiers $j \in \{1, \ldots, J_{k_n}\}$ la distance euclidienne entre les points $\mathbf{x}_{l_j}^n$ et $\mathbf{y}_j^{k_n}$ qu'on note :

$$d_j = \|\mathbf{x}_{l_j}^n - \mathbf{y}_j^{k_n}\|.$$

De plus, dans le cas où le point $\mathbf{x}_l^n$ est encadré par deux projetés consécutifs $\mathbf{x}_{l_j}^n$ et $\mathbf{x}_{l_{j+1}}^n$, c'est-à-dire quand $l_j < l < l_{j+1}$, les distances géodésiques s_j et s_{j+1} sur Γ_n entre ces différents points s'écrivent :

$$\begin{cases} s_j = \displaystyle\sum_{\tilde{l}=l_j}^{l-1} \|\mathbf{x}_{\tilde{l}+1}^n - \mathbf{x}_{\tilde{l}}^n\| \\ \\ s_{j+1} = \displaystyle\sum_{\tilde{l}=l}^{l_{j+1}-1} \|\mathbf{x}_{\tilde{l}+1}^n - \mathbf{x}_{\tilde{l}}^n\|. \end{cases}$$

Sous l'hypothèse d'ordonnancement précédente, plusieurs choses peuvent alors survenir :

- ou bien il existe $j \in \{1, \ldots, J_{k_n}\}$ tel que $l = l_j$ auquel cas on pose simplement

$$\mu(\mathbf{x}_l^n) = \frac{d_j}{1 + d_j};$$

- ou bien il existe $j \in \{1, \ldots, J_{k_n} - 1\}$ tel que $l_j < l < l_{j+1}$ et il vient (cf. Figure 16)

$$\mu(\mathbf{x}_l^n) = \frac{s_j}{s_{j+1} + s_j} \frac{d_{j+1}}{1 + d_{j+1}} + \frac{s_{j+1}}{s_{j+1} + s_j} \frac{d_j}{1 + d_j};$$

– ou bien $l < l_1$ et on a

$$\mu(\mathbf{x}_l^n) = \frac{d_1}{1+d_1};$$

– ou bien $l > l_{J_{k_n}}$ et il vient

$$\mu(\mathbf{x}_l^n) = \frac{d_{J_{k_n}}}{1+d_{J_{k_n}}};$$

– ou bien $l = l_j = l_{j+1} = \ldots = l_{j+J}$, ce qui correspond à la situation pathologique où $J+1$ points imposés ont le même projeté, auquel cas on pose

$$\mu(\mathbf{x}_l^n) = \frac{1}{J+1}\sum_{r=0}^{J}\frac{d_{j+r}}{1+d_{j+r}}.$$

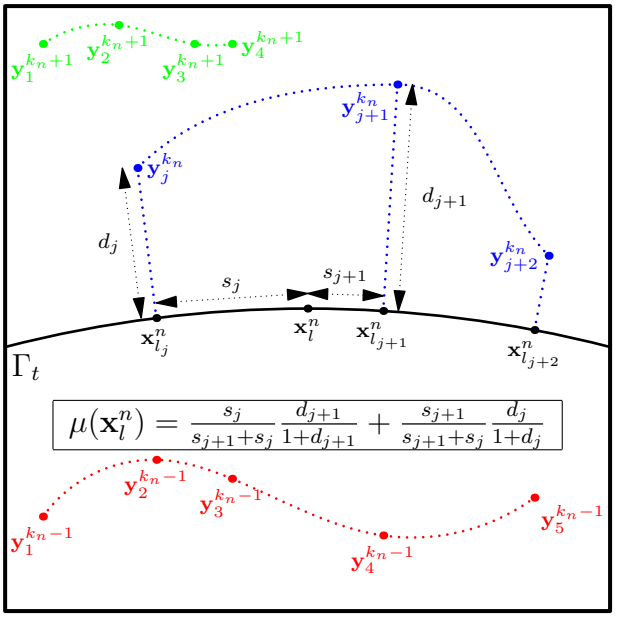

FIGURE 16 – *Schéma illustrant le calcul de* μ.

De manière moins formelle, la valeur de $\mu(\mathbf{x}_l^n) = \mu_l^n$ retenue est une combinaison linéaire convexe des précédents μ, les coefficients étant proportionnels à la distance géodésique entre le point et les projetés qui l'encadrent au mieux. Maintenant qu'on dispose des $(\mu_l^n)_{1\leqslant l\leqslant L}$ sur la courbe Γ_n, il convient de les calculer en chaque point de la grille $\mathbf{x}_{p,q}$. Et pour cela, on considère simplement la valeur du projeté orthogonal, c'est-à-dire qu'on pose :

$$\mu_{p,q}^n = \mu(\mathbf{x}_{p,q}^n) = \mu(p_{\Gamma_n}(\mathbf{x}_{p,q}^n)).$$

Ainsi, au fur et à mesure que le temps passe, la surface Γ_n va se rapprocher des points imposés et les valeurs de μ auront tendance à devenir nulles. Lorsque le régime stationnaire est atteint, on considère la couche $k_n + 1$ et si $k_n = K$, alors on pose $\mu = 1$.

Par ailleurs, on rappelle disposer de I orientations connues notées $(\vec{\mathbf{a}}_i)_{1\leqslant i\leqslant I}$ qui influent respectivement à l'intérieur de cercles de centre $\mathbf{x}_i$ et de rayon r_i calibré de telle sorte que les disques d'influence soient deux à deux disjoints. On commence par calculer la valeur des λ_i sur la surface Γ_n discrétisée en L points $(\mathbf{x}_l^n)_{1\leqslant l\leqslant L}$. Pour cela, on va ici choisir une expression pour les λ_i qui soit anisotrope contrairement à ce qui avait été fait auparavant. On espère ainsi mieux prendre en compte l'influence des vecteurs. On pose donc $\lambda_0(\mathbf{x}_l^n) = 1 - \sum_{i=1}^{I} \lambda_i(\mathbf{x}_l^n)$ où pour tout entier $i \in \{1, \ldots, I\}$ on a (cf. Figure 17) :

$$\lambda_i(\mathbf{x}_l^n) = \begin{cases} e^{\frac{(\vec{\mathbf{a}}_i \cdot (\mathbf{x}_l^n - \mathbf{x}_i))^2}{\left(\vec{\mathbf{a}}_i \cdot (\mathbf{x}_l^n - \mathbf{x}_i)\right)^2 - r_i^2}} & \text{si} \quad \|\mathbf{x}_l^n - \mathbf{x}_i\| < r_i \\ \\ 0 & \text{sinon} \end{cases}$$

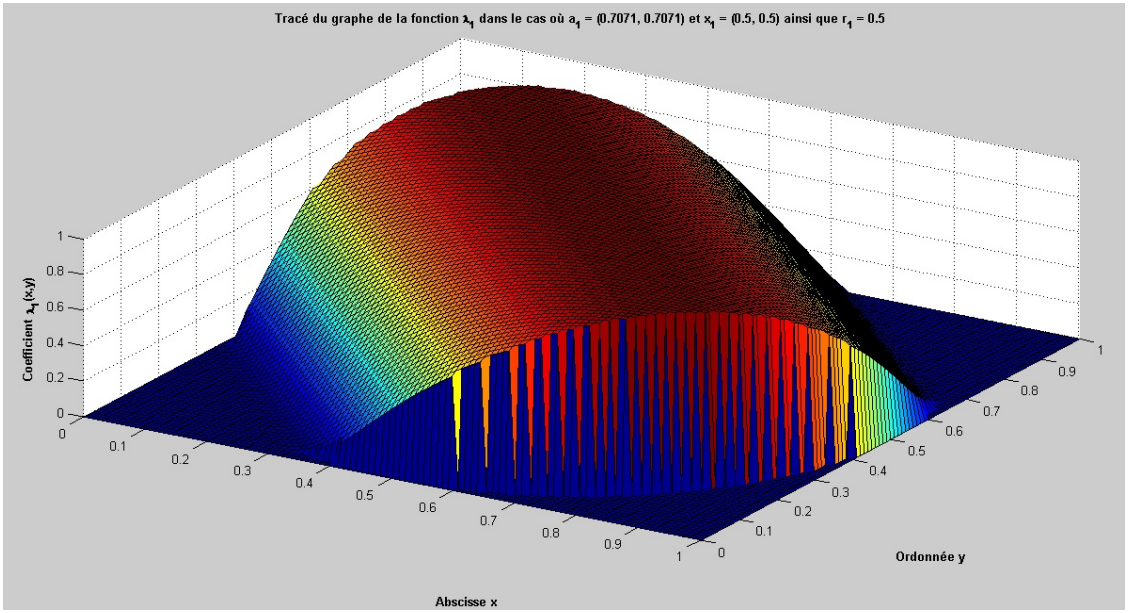

FIGURE 17 – *Tracé du graphe de λ_1 dans un cas 2-D avec $\vec{a}_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ et $\mathbf{x}_1 = (\frac{1}{2}, \frac{1}{2})$ ainsi qu'un rayon d'influence $r_1 = \frac{1}{2}$.*

Bien que les expressions choisies pour les λ_i ne soient plus continues, on verra qu'elles améliorent nettement les résultats obtenus pour les vecteurs, bien que ceux-ci ne soient pas encore complètement satisfaisants. Pour finir, une fois qu'on dispose des $\lambda_{i,l}^n = \lambda_i(\mathbf{x}_l^n)$, on les calcule aux points $\mathbf{x}_{p,q}$ de la grille en prenant comme valeur celle du projeté orthogonal, comme cela avait été fait pour μ :

$$\forall i \in \{1, \ldots, I\}, \forall (p,q) \in \{1, \ldots, P\} \times \{1, \ldots, Q\}, \quad \lambda_i(\mathbf{x}_{p,q}) = \lambda_i \left(p_{\Gamma_n}(\mathbf{x}_{p,q})\right).$$

3.2.4 Quatrième étape : évolution du flot grâce à un schéma de type Roe-Fix

Une fois les paramètres λ_i et μ correctement évalués en chaque point $\mathbf{x}_{p,q}$ de la grille, il convient également de le faire pour $\vec{\nabla}\Psi^n$ afin de pouvoir calculer la loi H^n au temps t_n. On pose alors pour tout entier $p \in \{1,\dots,P\}$ et tout $q \in \{1,\dots,Q\}$:

$$\left\{\begin{array}{l} \partial_x^+\Psi^n_{p,q} = \dfrac{\Psi^n_{p+1,q}-\Psi^n_{p,q}}{x_{p+1}-x_p} \\ \\ \partial_x^0\Psi^n_{p,q} = \dfrac{\Psi^n_{p+1,q}-\Psi^n_{p-1,q}}{x_{p+1}-x_{p-1}} \\ \\ \partial_x^-\Psi^n_{p,q} = \dfrac{\Psi^n_{p,q}-\Psi^n_{p-1,q}}{x_p-x_{p-1}} \end{array}\right. \quad \text{et} \quad \left\{\begin{array}{l} \partial_y^+\Psi^n_{p,q} = \dfrac{\Psi^n_{p,q+1}-\Psi^n_{p,q}}{y_{q+1}-y_q} \\ \\ \partial_y^0\Psi^n_{p,q} = \dfrac{\Psi^n_{p,q+1}-\Psi^n_{p,q-1}}{y_{q+1}-y_{q-1}} \\ \\ \partial_y^-\Psi^n_{p,q} = \dfrac{\Psi^n_{p,q}-\Psi^n_{p,q-1}}{y_q-y_{q-1}}. \end{array}\right.$$

On rappelle que Ψ^n est une distance signée et donc que $\|\vec{\nabla}\Psi\| = 1$. On cherche maintenant à faire évoluer le flot Ψ au temps t_{n+1} grâce à l'équation de Hamilton-Jacobi. On utilise un schéma d'Euler explicite en temps et un schéma Roe-fix en espace qui combine un schéma de type upwind et Lax-Friedrich, comme cela est indiqué dans [4]. En notant $\vec{\mathbf{a}}_i = (a_i^x, a_i^y)$ pour tout $i \in \{1,\dots,I\}$, on a pour tout entier $p \in \{1,\dots,P\}$ et tout $q \in \{1,\dots,Q\}$:

$$\left\{\begin{array}{l} \dfrac{\Psi^{n+1}_{p,q}-\Psi^n_{p,q}}{\Delta t} + H^n_{p,q} - \alpha_x^n \dfrac{\partial_x^+\Psi^n_{p,q}-\partial_x^-\Psi^n_{p,q}}{2} - \alpha_y^n \dfrac{\partial_y^+\Psi^n_{p,q}-\partial_y^-\Psi^n_{p,q}}{2} = 0. \\ \\ H^n_{p,q} = \mu^n_{p,q}\left[\lambda_0^n(\mathbf{x}_{p,q}) + \displaystyle\sum_{i=1}^{I} \lambda_i^n(\mathbf{x}_{p,q})\left(a_i^x \partial_x^\bigstar \Psi^n_{p,q} + a_i^y \partial_y^\bigstar \Psi^n_{p,q}\right)\right] \end{array}\right.$$

avec $\bigstar \in \{-,0,+\}$ ainsi que α_x^n et α_y^n qui suivent des règles qu'on va maintenant préciser. Pour cela, on introduit les quantités suivantes :

$$\left\{\begin{array}{l} H_x^\pm(\mathbf{x}_{p,q},t_n) = \mu^n_{p,q}\left(\lambda_0^n(\mathbf{x}_{p,q})\partial_x^\pm\Psi^n_{p,q} + \displaystyle\sum_{i=1}^{I}\lambda_i^n(\mathbf{x}_{p,q})a_i^x\right) \\ \\ H_y^\pm(\mathbf{x}_{p,q},t_n) = \mu^n_{p,q}\left(\lambda_0^n(\mathbf{x}_{p,q})\partial_y^\pm\Psi^n_{p,q} + \displaystyle\sum_{i=1}^{I}\lambda_i^n(\mathbf{x}_{p,q})a_i^y\right). \end{array}\right.$$

On distingue alors plusieurs cas :
- ou bien $H_x^+H_x^- \geqslant 0$ avec $H_x^+ < 0$ et alors $\bigstar = +$ ainsi que $\alpha_x^n = 0$;
- ou bien $H_x^+H_x^- \geqslant 0$ avec $H_x^+ \geqslant 0$ et alors $\bigstar = -$ ainsi que $\alpha_x^n = 0$;
- ou bien $H_x^+H_x^- < 0$ et alors $\bigstar = 0$ ainsi que

$$\alpha_x^n = \max_{1\leqslant p,q\leqslant P,Q}(H_x^+(\mathbf{x}_{p,q},t_n), H_x^-(\mathbf{x}_{p,q},t_n)).$$

On raisonne de la même manière avec $H_y^\pm$ pour déterminer les choix de α_y^n et de $\bigstar$ qui concerne la deuxième variable. Par conséquent, on obtient bien la valeur de flot Ψ au temps t_{n+1} en chaque point $\mathbf{x}_{p,q}$ de la grille.

Un algorithme de recherche de contour permet alors de retrouver $\Gamma_{n+1} = \Gamma_{t_{n+1}}$ et il est alors possible de réinitialiser Ψ^{n+1} en distance signée par la même méthode que celle décrite dans la deuxième étape, c'est-à-dire en posant

$$\Psi_{p,q}^{n+1} = \pm \Psi^{n+1}(p_{\Gamma_{t_{n+1}}}(\mathbf{x}_{p,q}))$$

le signe étant choisi positif si on est à l'extérieur de $\Gamma_{t_{n+1}}$ et négatif sinon (cf. Figure 10). Pour le savoir, il suffit de tracer une demi-droite verticale partant du point $\mathbf{x}_{p,q}$ considéré et allant vers l'intérieur de $\Gamma_{t_{n+1}}$. Si le nombre d'intersection entre cette dernière et Γ_{n+1} est impair (respectivement pair), alors c'est que le point $\mathbf{x}_{p,q}$ est à l'extérieur (respectivement à l'intérieur) et on choisira un signe positif (respectivement négatif).

3.2.5 Résultats obtenus

Comme on peut le voir sur les Figures 18 et 19, les résultats sont bien meilleurs que ceux obtenus lors de l'approche lagrangienne précédente. L'algorithme prend très bien en compte les contraintes ponctuelles mais concernant les vecteurs, l'orientation finale n'est pas encore suffisamment satisfaisante.

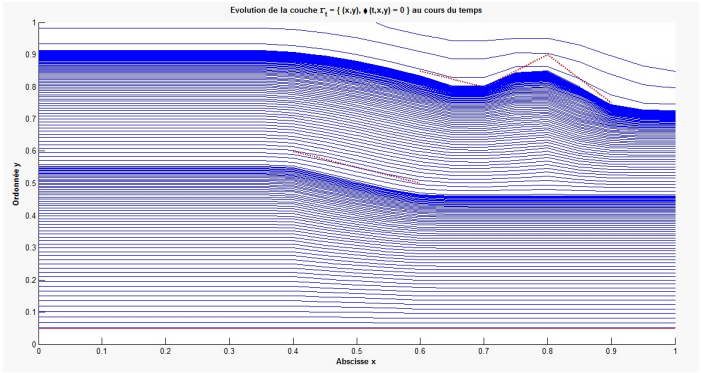

FIGURE 18 – *Exemple de reconstitution 2-D de sous-sol à partir de plusieurs points.*

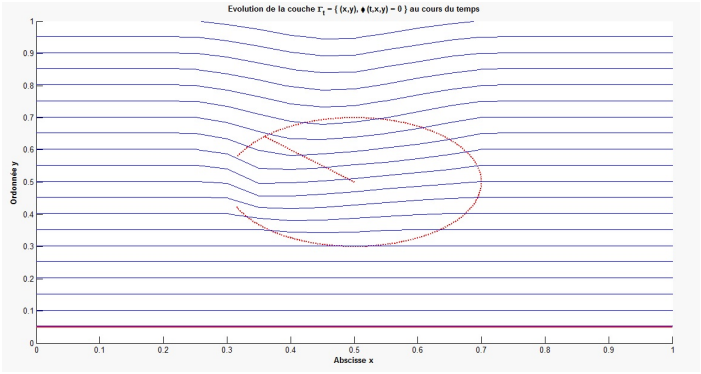

FIGURE 19 – *Exemple de reconstitution 2-D de sous-sol à partir d'un vecteur imposé.*

Conclusion

Au cours de cette Semaine d'Etude Maths-Entreprises, le consortium pétrolier GOCAD nous a proposé de réfléchir sur les différentes manières de reconstruire le sous-sol terrestre à partir de données discrètes éparses.

Après avoir pris connaissance d'un certain nombre d'articles, notamment de l'approche statistique effectuée dans [2] ainsi que de la démarche géologique et numérique développée dans [3], nous nous sommes attardés sur le point de vue théorique adopté dans [1] et avons tenté de l'adapter à notre problématique.

Le modèle numérique obtenu grâce à [4] permet de reconstruire correctement le sous-sol en un temps raisonnable. L'algorithme utilisé semble prometteur, notamment l'approche lagrangienne qui permettrait de passer à la 3-D rapidement.

Références

[1] Ø. Hjelle and S. A. Petersen, *A Hamilton-Jacobi framework for modeling folds in structural geology*, Mathematical Geosciences, **43** (7), 741-761, October 2011.

[2] P. Calcagno, J.-P. Chils, G. Courrioux and A. Guillen, *Geological modelling from field data and geological knowledge Part I. Modelling method coupling 3D potential-field interpolation and geological rules*, Physics of the Earth and Planetary Interiors, **171** (1-4), 147-157, December 2008.

[3] G. Caumon, G. Gray, C. Antoine and M.-O. Titeux, *3D Implicit Stratigraphic Model Building From Remote Sensing Data on Tetrahedral Meshes : Theory and Application to a Regional Model of La Popa Basin, NE Mexico*, IEEE Transactions on Geoscience and Remote Sensing, **51** (3), 1613-1621, March 2013.

[4] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Applied Mathematical Sciences, **153**, Springer, 2003.

Annexe : les progammes MATLAB

L'approche lagrangienne

```
clear all
close all


% Initialisation des alertes
alerte_vecteurs = 0;
alerte_points = 0;
alerte_couches = 0;


% Données vectorielles a_i = [x_ai ai_x; y_ai ai_y]
% Sinon poser a1 = [1 0; 1 0];
a1 = [.5 .2/sqrt(2); .35 .2/sqrt(2)];
r1 = norm(a1(:,2));


if r1==0
alerte_vecteurs = 1;
else
a1(:,2) = a1(:,2)/r1;
```

```
end


% Données ponctuelles pour la k-ième couche [xk_1 ... xk_n ; yk_1 ... yk_n]
% Sinon poser x1 = nulle ;
nulle = [0 ; 0] ;
x1 = [.1 .15 .2 .25 .3 .35 ; .1 .15 .2 .2 .15 .1] ;
x2 = [.6 .7 .8 .9 ; .9 .8 .6 .8] ;


if max(max(abs(x1)))==0
alerte_points = 1 ;
else
x0 = [x1 nulle x2 nulle] ;
i0 = find(x0(2, :)==0) ;
points = [[1 ; i0(1)-1] x0( : , 1 : i0(1)-1)] ;
end


% Vecteur spatio-temporel
dt = 1e-4 ;
tmax = .55 ;
T = 0 : dt : tmax ;
lT = length(T) ;
t_graph = 400 ;


dx = 1e-2/4 ;
xmin = 0 ;
xmax = 1 ;
X = xmin : dx : xmax ;
lX = length(X) ;


% Initialisation des données sur la surface Gamma0 = [0,1]x{0}
Xdata = X' ;
Ydata = zeros(lX,1) ;
Pxdata = zeros(lX,1) ;
Pydata = ones(lX,1) ;


Xsurf = [Xdata' ; Ydata'] ;
Psurf = [Pxdata' ; Pydata'] ;


lambda0 = ones(1,lX) ;
lambda1 = zeros(1,lX) ;
mu = ones(1,lX) ;


dlambda1 = zeros(2,lX) ;
dlambda0 = zeros(2,lX) ;
dmu = zeros(2,lX) ;


% Tracé des données initiales
figure
hold on
plot(Xdata( : ,1),Ydata( : ,1),'r','linewidth',1.5)


xlabel('\bf{Abscisse x}')
ylabel('\bf{Ordonnée y}')
title('\bf{Reconstitution du sous-sol avec un vecteur a_1 = (1,1), x_1=(0.5,0.35) et r_1=0.2}')


if alerte_points==0
plot(x1(1, :),x1(2, :),'r :','linewidth',1.5)
plot(x2(1, :),x2(2, :),'r :','linewidth',1.5)
end


if alerte_vecteurs==0
plot([a1(1,1) a1(1,1)+r1*a1(1,2)],[a1(2,1) a1(2,1)+r1*a1(2,2)],'r :','linewidth',1.5')
plot(a1(1,1) + r1*cos([-7*pi/8 : 1e-2 : 7*pi/8]),a1(2,1)+r1*sin([-7*pi/8 : 1e-2 : 7*pi/8]), 'r :','linewidth',1.5)
end


% Boucle temporelle
for n = 1 : lT-1
% On effectue les actions suivantes s'il y a des vecteurs imposés
if alerte_vecteurs==0
% Calcul des lambdas sur la surface
a = ((Xsurf(1, :) - a1(1,1)).^2 + (Xsurf(2, :) - a1(2,1)).^2)/r1^2 ;
lambda1 = exp(1+(a<1)./(a-1+(a==1))).*(a<1) ;
lambda0 = 1-lambda1 ;


% Calcul des gradients de lambda
dlambda1=(-2/r1^2).*lambda1./(a-1+(a==1)).^2 ;
dlambda1 = (Xsurf - a1( : ,1)*ones(1,lX)).*(ones(2,1)*dlambda1) ;
dlambda0 = -dlambda1 ;
end
```

```
    % On effectue les actions suivantes s'il y a des points imposés
    if alerte_points==0
    % On calcule la distance entre les points imposés et leurs projetés
    distance = sqrt((Xsurf(1, :)' * ones(1,points(2,1)) - ones(lX,1) * points(1,2 : end)).^2 + (Xsurf(2, :)' * ones(1,points(2,1)) -
ones(lX,1) * points(2,2 : end)).^2);
    min_dist = min(distance);


    % On ré-initialise la couche si besoin
    if max(min_dist)<dx
    if points(1,1)==length(i0)
    alerte_couches = 1;
    else
    points = [[points(1,1) + 1; i0(points(1,1) + 1) - i0(points(1,1)) - 1] x0( : ,i0(points(1,1)) + 1 : i0(points(1,1) + 1) - 1)];
    distance = sqrt((Xsurf(1, :)' * ones(1,points(2,1)) - ones(lX,1) * points(1,2 : end)).^2 + (Xsurf(2, :)' * ones(1,points(2,1)) -
ones(lX,1) * points(2,2 : end)).^2);
    min_dist = min(distance);
    end
    end


    % On retrouve à quels indices les projetés correspondent
    indice = ones(1,points(2,1));
    for k=1 : points(2,1)
    indice(k) = min(find(distance( : ,k)==min_dist(k)));
    end


    % Calcul de mu
    if alerte_couches==1
    mu = ones(1,lX);
    dmu = zeros(2,lX);
    else
    mu = (min_dist(1)/(1 + min_dist(1)))*ones(1,lX);
    mu(indice(end)+1 : end) = min_dist(end)/(1 + min_dist(end));
    for k=2 : points(2,1)
    dm = min_dist(k-1)/(1 + min_dist(k-1));
    dp = min_dist(k)/(1+min_dist(k));
    if indice(k-1)==indice(k)
    mu(indice(k)) = (dm + dp)/2;
    else
    sm = cumsum(sqrt((Xsurf(1,indice(k-1)+1 : indice(k)) - Xsurf(1,indice(k-1) : indice(k)-1)).^2 + (Xsurf(2,indice(k-1)+1 : in-
dice(k)) - Xsurf(2,indice(k-1) : indice(k)-1)).^2));
    sp = sm(end) - sm;
    mu(indice(k-1)+1 : indice(k)) = (dm - dp)*(sm<=sp) + dp;
    end
    end


    % Calcul du gradient de mu
    dmu( : ,1) = (mu(2) - mu(1))*(Xsurf( : ,2)-Xsurf( : ,1))/norm(Xsurf( : ,2)-Xsurf( : ,1)+(Xsurf( : ,2)==Xsurf( : ,1)));
    aa = sqrt((Xsurf(1,2 : end) - Xsurf(1,1 : end-1)).^2 + (Xsurf(2,2 : end) - Xsurf(2,1 : end-1)).^2);
    aa = (mu(2 : end) - mu(1 : end-1))./(aa + (aa==0));
    dmu( : ,2 : end) = (ones(2,1)*aa).*(Xsurf( : ,2 : end)-Xsurf( : ,1 : end-1));
    end
    end


    % Ré-initialisation des couches
    normeP = sqrt(Psurf(1, :).^2 + Psurf(2, :).^2);
    a1pointP = Psurf(1, :)*a1(1,2) + Psurf(2, :)*a1(2,2);
    Xsurf = Xsurf + dt*(ones(2,1)*mu).*((ones(2,1)*(lambda0./(normeP + (normeP==0)))).*Psurf + a1( :,2)*lambda1);
    Psurf = Psurf - dt * dmu .* (ones(2,1) * (lambda0 .* normeP + lambda1 .* a1pointP)) - dt * (ones(2,1) * mu) .* (dlambda0 .*
(ones(2,1) * normeP) + dlambda1 .* (ones(2,1) * a1pointP));


    % Stockage des données et tracé du graphique
    if mod(n,t_graph)==0
    Xdata = [Xdata Xsurf(1, :)'];
    Ydata = [Ydata Xsurf(2, :)'];


    Pxdata = [Pxdata Psurf(1, :)'];
    Pydata = [Pydata Psurf(2, :)'];


    plot(Xsurf(1, :),Xsurf(2, :),'linewidth',1.5)
    end

    end
```

L'approche eulérienne

```
clear all
close all


% Initialisation des alertes
alerte_couches = 0;
alerte_points = 0;
alerte_vecteurs = 0;
alerte_schemas = 0;


% Données numériques
dx = 1e-1/2;
x_min = 0;
x_max = 1;


dy = dx;
y_min = x_min;
y_max = x_max;


dt = 1e-2/4;
t_max = 200;
t_graph = 1e-1;
tol = min(dx,dy);


% Vecteurs imposés mis sous la forme a_i=[x_i ai_x; y_i ai_y]
% Pour n'en mettre aucun, poser a1 = [1 0; 1 0];
a1 = [.35 .2*cos(pi/8); .5 .2*sin(pi/8)];


r1 = norm(a1(:,2));
if r1==0
alerte_vecteurs = 1;
else
a1(:,2) = a1(:,2)/r1;
end


% Points imposés sur la k-ième couches x_k=[xk_1 xk_j; yk_1 yk_j]
% Pour n'en mettre aucun, poser x1 = nulle;
nulle = [0; 0];
x1 = [.4 .5 .6; .6 .55 .5];
x2 = [.6 .7 .8 .9; .85 .8 .9 .75];


if max(max(abs(x1)))==0
alerte_points = 1;
else
x0 = [x1 nulle x2 nulle];
i0 = find(x0(2,:)==0);
point_impose = [[1; i0(1) - 1] x0(:,1 : i0(1) - 1)];
end


% Création des vecteurs spatiaux
X = x_min : dx : x_max;
lX = length(X);
Y = y_min : dy : y_max;
lY = length(Y);
[XX YY] = meshgrid(X,Y);


% Initialisation de Gamma0 par la droite y = dy (prendre phi(0,x,y) = y - dy)
PHI = YY - dy;


% Initialisation de la matrice signe (+1 ext = au dessus, -1 int = en dessous)
signe = ones(lY,lX);
signe(:,end) = -1;


% Initialisation du gradient de PHI
dPHIx_plus = zeros(lY,lX);
dPHIx_moins = zeros(lY,lX);
dPHIy_moins = ones(lY,lX);
dPHIy_plus = ones(lY,lX);


alphaX = zeros(lY,lX);
alphaY = zeros(lY,lX);


% Initialisation des paramètres
MU = ones(lY,lX);
lambda0 = ones(lY,lX);
LAMBDA1 = zeros(lY,lX);


% Tracé des données initiales
figure
```

```
hold on
plot(X, dy + 0*X,'r','linewidth',1.5);


if alerte_vecteurs==0
plot([a1(1,1) a1(1,1) + r1*a1(1,2)], [a1(2,1) a1(2,1) + r1*a1(2,2)],'r :','linewidth',1.5')
plot(a1(1,1) + r1*cos([-7*pi/8 : 1e-2 : 7*pi/8]), a1(2,1) + r1*sin([-7*pi/8 : 1e-2 : 7*pi/8]),'r :','linewidth',1.5')
end


if alerte_points==0
plot(x0(1,1 : i0(1) - 1), x0(2,1 : i0(1) - 1),'r :','linewidth',1.5')
plot(x0(1,i0(1) + 1 : i0(2) - 1), x0(2,i0(1) + 1 : i0(2) - 1),'r :','linewidth',1.5')
end


xlabel('\bf{Abscisse x}')
ylabel('\bf{Ordonnée y}')
title('\bf{Evolution de la couche \Gamma_t = \{ (x,y), \phi (t,x,y) = 0 \} au cours du temps}')


% Boucle temporelle
for t=1 :dt :t_max
% Calcul du gradient de PHI par différences finies décentrées droite/gauche
dPHIx_plus( :,1 : end-1) = (PHI( :,2 : end) - PHI( :,1 : end-1))/dx;
dPHIx_moins( :,2 : end) = dPHIx_plus( :,1 : end-1);
dPHIx_plus( :,end) = dPHIx_moins( :,end);
dPHIx_moins( :,1) = dPHIx_plus( :,1);


dPHIy_plus(1 : end-1, :) = (PHI(2 : end, :) - PHI(1 : end-1, :))/dx;
dPHIy_moins(2 : end, :) = dPHIy_plus(1 : end-1, :);
dPHIy_plus(end, :) = dPHIy_moins(end, :);
dPHIy_moins(1, :) = dPHIy_plus(1, :);


% Calcul des points de la surface PHI=0 mis sous la forme [0 x ; nbre2pts y]
point_surface=contourc(X,Y,PHI,[0,0]);


% On effectue les actions suivantes s'il y a des points imposés
if alerte_points==0
% On gère le numéro des couches en modifiant point_impose si nécessaire
distance = sqrt((point_surface(1, 2 : end)' * ones(1, point_impose(2,1)) - ones(point_surface(2,1), 1) * point_impose(1, 2 :
end)).^2 + (point_surface(2, 2 : end)' * ones(1, point_impose(2,1)) - ones(point_surface(2,1), 1) * point_impose(2, 2 : end)).^2);
min_dist = min(distance);
if max(min_dist)<tol
if point_impose(1,1)==length(i0)
alerte_couches = 1;
else
point_impose = [[point_impose(1,1)+1 ; i0(point_impose(1,1) + 1) - i0(point_impose(1,1)) - 1] x0( : ,i0(point_impose(1,1))
+ 1 : i0(point_impose(1,1) + 1) -1)];
distance = sqrt((point_surface(1, 2 : end)' * ones(1, point_impose(2,1)) - ones(point_surface(2,1), 1) * point_impose(1, 2 :
end)).^2 + (point_surface(2, 2 : end)' * ones(1, point_impose(2,1)) - ones(point_surface(2,1), 1) * point_impose(2, 2 : end)).^2);
min_dist = min(distance);
end
end


% Calcul de mu
if alerte_couches==1
MU = ones(lY,lX);
else
% On retrouve à quels points les projetés correspondent
indice = ones(1,point_impose(2,1));
for k=1 : point_impose(2,1)
indice(k) = min(find(distance( : ,k)==min_dist(k)));
end


% Calcul de mu sur la surface
mu = (min_dist(1)/(1 + min_dist(1))) * ones(1,point_surface(2,1));
mu(indice(end) + 1 : end) = min_dist(end)/(1 + min_dist(end));
for k = 2 : point_impose(2,1)
dm = min_dist(k - 1)/(1 + min_dist(k - 1));
dp = min_dist(k)/(1 + min_dist(k));
if indice(k - 1)>=indice(k)
mu(indice(k)) = (dm + dp)/2;
else
sm = cumsum(sqrt((point_surface(1,indice(k - 1) + 2 : indice(k) + 1) - point_surface(1,indice(k - 1) + 1 : indice(k))).^2 +
(point_surface(2,indice(k - 1) + 2 : indice(k) + 1) - point_surface(2,indice(k - 1) + 1 : indice(k))).^2));
sp = sm(end) - sm;
mu(indice(k - 1) + 1 : indice(k)) = (dm*sp + dp*sm)./(sp + sm);
end
end


% Calcul de mu sur tout le domaine
d1 = (X'*ones(1,point_surface(2,1)) - ones(lX,1)*point_surface(1,2 : end)).^2;
d2 = (Y'*ones(1,point_surface(2,1)) - ones(lY,1)*point_surface(2,2 : end)).^2;
for i=1 : lX
for j=1 : lY
d = sqrt(d1(i, :) + d2(j, :));
```

```
i_min = min(find(d==min(d)));
MU(j,i) = mu(i_min);
end
end
end
end


% On effectue les actions suivantes s'il y a des vecteurs imposés
if alerte_vecteurs==0;
% Calcul de lambda1 sur la surface
dist1 = ((point_surface(1,2 : end) - a1(1,1)).^2 + (point_surface(2,2 : end) - a1(2,1)).^2)/r1^2;
dist2 = ((point_surface(1,2 : end) - a1(1,1))*a1(1,2) + (point_surface(2,2 : end) - a1(2,1))*a1(2,2)).^2/r1^2;
lambda1 = exp(1 + (dist2<1)./(dist2 - 1 + (dist2==1))).*(dist2<1).*(dist1<1);


% Calcul de lambda1 sur tout le domaine
dd1 = (X'*ones(1,point_surface(2,1)) - ones(lX,1)*point_surface(1,2 : end)).^2;
dd2 = (Y'*ones(1,point_surface(2,1)) - ones(lY,1)*point_surface(2,2 : end)).^2;
for i=1 : lX
for j=1 : lY
dd = sqrt(dd1(i, :) + dd2(j, :));
i_min1 = min(find(dd==min(dd)));
LAMBDA1(j,i) = lambda1(i_min1);
end
end
lambda0=1-LAMBDA1;
end


% Détermination du schéma Roe-Fix=Upwind+LaxFriedrichs
if alerte_schemas==1;
dPHIx = dPHIx_moins;
dPHIy = dPHIy_moins;
else
% Calcul des dérivées partielles de H et des coefficients dissipatifs
H1m = MU.*(lambda0.*dPHIx_moins + LAMBDA1*a1(1,2));
H1p = MU.*(lambda0.*dPHIx_plus + LAMBDA1*a1(1,2));
H2m = MU.*(lambda0.*dPHIy_moins + LAMBDA1*a1(2,2));
H2p = MU.*(lambda0.*dPHIy_plus + LAMBDA1*a1(2,2));


alphaX0 = max(max(max(abs(H1m))),max(max(abs(H1p))));
alphaY0 = max(max(max(abs(H2m))),max(max(abs(H2p))));


H1 = (H1m.*H1p>=0);
Hx = (H1p<0);
dPHIx = H1.*(dPHIx_plus.*Hx + dPHIx_moins.*(1 - Hx)) + (1 - H1).*(dPHIx_plus + dPHIx_moins)/2;
alphaX = alphaX0*(1 - H1);


H2 = (H2m.*H2p>=0);
Hy = (H2p<0);
dPHIy = H2.*(dPHIy_plus.*Hy + dPHIy_moins.*(1 - Hy)) + (1 - H2).*(dPHIy_plus + dPHIy_moins)/2;
alphaY = alphaY0*(1 - H2);
end


% Calcul du nouveau PHInew = 0 par l'équation PHI + MU*(lambda0 + lambda1*(dPHI.a1)) = 0
PHInew = PHI - dt * (MU .* (lambda0 + LAMBDA1 .* (a1(1,2) * dPHIx + a1(2,2) * dPHIy)) - alphaX .* (dPHIx_plus-
dPHIx_moins) / 2 - alphaY .* (dPHIy_plus - dPHIy_moins) / 2);
surf_new=contourc(X,Y,PHInew,[0,0]);


% Ajustage du nouveau phi en distance signée
d3 = (X'*ones(1,surf_new(2,1)) - ones(lX,1)*surf_new(1,2 : end)).^2;
d4 = (Y'*ones(1,surf_new(2,1)) - ones(lY,1)*surf_new(2,2 : end)).^2;
for i=1 : lX
for j=1 : lY
% Calcul PHI(x)=distance(x,surf_new)
PHI(j,i) = min(sqrt(d3(i, :) + d4(j, :)));


% Evaluation du signe
cond = (surf_new(1,2 : end) < XX(j,i) + dx/2).*(surf_new(1,2 : end)> XX(j,i) - dx/2).*(surf_new(2,2 : end) < YY(j,i));
n = sum(abs(diff(cond)))/2;
if mod(n,2)==0
signe(j,i) = -1;
else
signe(j,i) = 1;
end
end
end
PHI = PHI.*signe;


% Tracé des graphiques
if mod(t,t_graph)==0
contour(XX,YY,PHInew,[0,0]);
end

end
```